

---

## 28-Pin, Low-Power, High-Performance Microcontrollers with CAN Technology

---

### Description

The PIC18(L)FXXK83 is a full-featured CAN product family that can be used in automotive and industrial applications. The multitude of communication peripherals found on the product family, such as CAN, SPI, two I<sup>2</sup>Cs, two UARTs, LIN, DMX, and DALI can handle a wide range of wired and wireless (using external modules) communication protocols for intelligent applications. This family includes a 12-bit ADC with Computation (ADC<sup>2</sup>) extensions for automated signal analysis to reduce the complexity of the application. This, combined with the Core Independent Peripherals integration capabilities, enables functions for motor control, power supply, sensor, signal and user interface applications.

### Core Features

- C Compiler Optimized RISC Architecture
- Operating Speed:
  - Up to 64 MHz clock operation
  - 62.5 ns minimum instruction cycle
- Two Direct Memory Access (DMA) Controllers:
  - Data transfers to SFR/GPR spaces from either Program Flash Memory, Data EEPROM or SFR/GPR spaces
  - User-programmable source and destination sizes
  - Hardware and software-triggered data transfers
- System Bus Arbiter with User-Configurable Priorities for Scanner and DMA1/DMA2 with respect to the main line and interrupt execution
- Vectored Interrupt Capability:
  - Selectable high/low priority
  - Fixed interrupt latency
  - Programmable vector table base address
- 31-Level Deep Hardware Stack
- Low-Current Power-on Reset (POR)
- Configurable Power-up Timer (PWRT)
- Brown-Out Reset (BOR)
- Low-Power BOR (LPBOR) Option
- Windowed Watchdog Timer (WWDT):
  - Variable prescaler selection
  - Variable window size selection
  - Configurable in hardware or software

### Memory

- Up to 64 KB Flash Program Memory
- Up to 4 KB Data SRAM Memory
- Up to 1 KB Data EEPROM
- Memory Access Partition (MAP):
  - Configurable boot and app region sizes with individual write-protections
- Programmable Code Protection
- Device Information Area (DIA) stores:
  - Unique IDs and Device IDs
  - Temp Sensor factory-calibrated data
  - Fixed Voltage Reference calibrated data
- Device Configuration Information (DCI) stores:
  - Erase row size
  - Number of write latches per row
  - Number of user rows
  - Data EEPROM memory size
  - Pin count

### Operating Characteristics

- Operating Voltage Range:
  - 1.8V to 3.6V (PIC18LF25/26K83)
  - 2.3V to 5.5V (PIC18F25/26K83)
- Temperature Range:
  - Industrial: -40°C to 85°C
  - Extended: -40°C to 125°C

### Power-Saving Functionality

- DOZE mode: Ability to run CPU core slower than the system clock
- IDLE mode: Ability to halt CPU core while internal peripherals continue operating
- SLEEP mode: Lowest power consumption
- Peripheral Module Disable (PMD):
  - Ability to disable unused peripherals to minimize power consumption

## eXtreme Low-Power (XLP) Features

- Sleep mode: 60 nA @ 1.8V, typical
- Windowed Watchdog Timer: 720 nA @ 1.8V, typical
- Secondary Oscillator: 580 nA @ 32 kHz
- Operating Current:
  - 4  $\mu$ A @ 32 kHz, 1.8V, typical
  - 45  $\mu$ A/MHz @ 1.8V, typical

## Digital Peripherals

- Three 8-Bit Timers (TMR2/4/6) with Hardware Limit Timer (HLT):
  - Hardware monitoring and Fault detection
- Four 16-Bit Timers (TMR0/1/3/5)
- Four Configurable Logic Cell (CLC):
  - Integrated combinational and sequential logic
- Three Complementary Waveform Generators (CWGs):
  - Rising and falling edge dead-band control
  - Full-bridge, half-bridge, 1-channel drive
  - Multiple signal sources
  - Programmable dead band
  - Fault-shutdown input
- Four Capture/Compare/PWM (CCP) modules
- Four 10-bit Pulse-Width Modulators (PWMs)
- Numerically Controlled Oscillator (NCO):
  - Generates true linear frequency control
  - High resolution using 20-bit accumulator and 20-bit increment values
- DSM: Data Signal Modulator:
  - Multiplex two carrier clocks, with glitch prevention feature
  - Multiple sources for each carrier
- Programmable CRC with Memory Scan:
  - Reliable data/program memory monitoring for fail-safe operation (e.g., Class B)
  - Calculate CRC over any portion of program memory or data EEPROM
- Two UART Modules:
  - Modules are asynchronous and compatible with RS-232 and RS-485
  - Support LIN Master and Slave, DMX mode, DALI Gear and Device protocols
  - Automatic and user-timed BREAK period generation
  - DMA Compatible
  - Automatic checksums
  - Programmable 1, 1.5, and two Stop bits
  - Wake-up on BREAK reception

- One SPI module:
  - Configurable length bytes
  - Configurable length data packets
  - Receive-without-transmit option
  - Transmit-without-receive option
  - Transfer byte counter
  - Separate Transmit and Receive Buffers with 2-byte FIFO and DMA capabilities
- CAN module:
  - Conforms to CAN 2.0B Active Specification
  - Three operating modes: Legacy (compatible with existing PIC18CXX8/FXX8 CAN modules), Enhanced mode, and FIFO mode.
  - Message bit rates up to 1 Mbps
  - DeviceNet™ data byte filter support
  - Six programmable receive/transmit buffers
  - Three dedicated transmit buffers
  - Two dedicated receive buffers
  - 16 Full, 29-bit acceptance filters with dynamic association
  - Three full, 29-bit acceptance masks
  - Automatic remote frame handling
  - Advanced error management features.
- Two I<sup>2</sup>C modules, SMBus, PMBus™ compatible:
  - Dedicated Address, Transmit and Receive buffers
  - Bus Collision Detection with arbitration
  - Bus time-out detection and handling
  - Multi-Master mode
  - Separate Transmit and Receive Buffers with 2-byte FIFO and DMA capabilities
  - I<sup>2</sup>C, SMBus 2.0 and SMBus 3.0, and 1.8V input level selections
  - Supports Standard-mode (100 kHz), Fast-mode (400 kHz) and Fast-mode plus (1 MHz) modes of operation
- Device I/O Port Features:
  - 25 I/O pins
  - One input-only pin (RE3)
  - Individually programmable I/O direction, open-drain, slew rate, weak pull-up control
  - Interrupt-on-change
  - Three External Interrupt Pins
- Peripheral Pin Select (PPS):
  - Enables pin mapping of digital I/O
- Two Signal Measurement Timer (SMT):
  - 24-bit timer/counter with prescaler

## Analog Peripherals

- Analog-to-Digital Converter with Computation (ADC<sup>2</sup>):
  - 12-bit with up to 24 external channels up to 140 ksp/s
  - Automated post-processing
  - Automated math functions on input signals: averaging, filter calculations, oversampling and threshold comparison
  - Operates in Sleep
  - Integrated charge pump for improved low-voltage operation
- Hardware Capacitive Voltage Divider (CVD):
  - Automates touch sampling and reduces software size and CPU usage when touch or proximity sensing is required
  - Adjustable sample and hold capacitor array
  - Two guard ring output drives
- Temperature Sensor:
  - Internal connection to ADC
  - Can be calibrated for improved accuracy
- Two Comparators:
  - Low-Power/High-Speed mode
  - Fixed Voltage Reference at noninverting input(s)
  - Comparator outputs externally accessible
- 5-Bit Digital-to-Analog Converter (DAC):
  - 5-bit resolution, rail-to-rail
  - Positive Reference Selection
  - Unbuffered I/O pin output
  - Internal connections to ADCs and comparators
- Voltage Reference:
  - Fixed Voltage Reference with 1.024V, 2.048V and 4.096V output levels

## Flexible Oscillator Structure

- High-Precision Internal Oscillator:
  - Selectable frequency range up to 64 MHz
  - $\pm 1\%$  at calibration (nominal)
- Low-Power Internal 32 kHz Oscillator (LFINTOSC)
- External 32 kHz Crystal Oscillator (SOCS)
- External Oscillator Block with:
  - x4 PLL with external sources
  - Three crystal/resonator modes up to 20 MHz
  - Three external clock modes up to 20 MHz
- Fail-Safe Clock Monitor
- Oscillator Start-up Timer (OST):
  - Ensures stability of crystal oscillator sources

# PIC18(L)F25/26K83

**TABLE 1: PIC18(L)FXXK83 FAMILY TYPES**

| Device         | Data Sheet Index | Program Flash Memory (KB) | Data EEPROM (B) | Data SRAM (bytes) | I/O Pins | 12-bit ADC <sup>2</sup> (ch) | 5-bit DAC | Comparator | 8-bit/ (with HLT)/16-bit Timer | Window Watchdog Timer (WWDT) | Signal Measurement Timer (SMT) | CCP/10-bit PWM | CWG | NCO | CLC | Zero-Cross Detect | Direct Memory Access (DMA) | Memory Access Partition | Vectored Interrupts | CAN | UART with Protocols | I <sup>2</sup> C/SPI | Peripheral Pin Select | Peripheral Module Disable | Debug <sup>(1)</sup> |
|----------------|------------------|---------------------------|-----------------|-------------------|----------|------------------------------|-----------|------------|--------------------------------|------------------------------|--------------------------------|----------------|-----|-----|-----|-------------------|----------------------------|-------------------------|---------------------|-----|---------------------|----------------------|-----------------------|---------------------------|----------------------|
| PIC18(L)F25K83 | (A)              | 32                        | 1024            | 2048              | 25       | 24                           | 1         | 2          | 3/4                            | Y                            | Y                              | 4/4            | 3   | 1   | 4   | Y                 | 2                          | Y                       | Y                   | Y   | 2                   | 2/1                  | Y                     | Y                         | I                    |
| PIC18(L)F26K83 | (A)              | 64                        | 1024            | 4096              | 25       | 24                           | 1         | 2          | 3/4                            | Y                            | Y                              | 4/4            | 3   | 1   | 4   | Y                 | 2                          | Y                       | Y                   | Y   | 2                   | 2/1                  | Y                     | Y                         | I                    |

**Note 1:** I - Debugging integrated on chip.

**Data Sheet Index:**

A: DS40001943 PIC18(L)F25/26K83 Data Sheet, 28-Pin

**Note:** For other small form-factor package availability and marking information, visit <http://www.microchip.com/packaging> or contact your local sales office.

# PIC18(L)F25/26K83

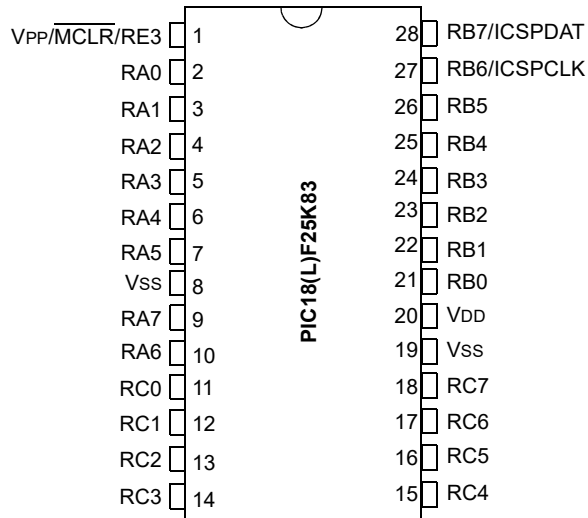
**TABLE 2: PACKAGES**

| Device         | SPDIP | SOIC | SSOP | UQFN | QFN |
|----------------|-------|------|------|------|-----|
| PIC18(L)F25K83 | •     | •    | •    | •    | •   |
| PIC18(L)F26K83 | •     | •    | •    | •    | •   |

**Note 1:** Pin details are subject to change.

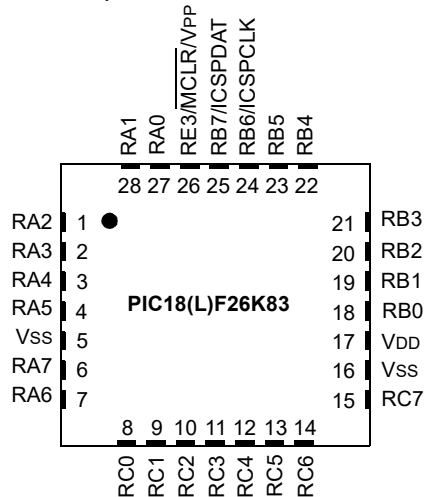
## Pin Diagrams

### 28-pin SPDIP, SOIC, SSOP



**Note:** See [Table 3](#) for location of all peripheral functions.

### 28-pin QFN (6x6x0.9mm), UQFN (4x4x0.5mm)



**Note 1:** See [Table 3](#) for location of all peripheral functions.

**2:** It is recommended that the exposed bottom pad be connected to VSS, however it must not be the only VSS connection to the device.

## Pin Allocation Tables

**TABLE 3: 28-PIN ALLOCATION TABLE (PIC18(L)F25/26K83)**

| I/O | 28-Pin SPDIP/SOIC/SSOP | 28-Pin (U)QFN | ADC                          | Voltage Reference | DAC      | Comparators      | Zero Cross Detect | I <sup>2</sup> C        | SPI                  | UART                | DSM                    | Timers/SMT                                                                                   | CCP and PWM         | CWG                 | CLC                   | NCO |
|-----|------------------------|---------------|------------------------------|-------------------|----------|------------------|-------------------|-------------------------|----------------------|---------------------|------------------------|----------------------------------------------------------------------------------------------|---------------------|---------------------|-----------------------|-----|
| RA0 | 2                      | 27            | ANA0                         | —                 | —        | C1IN0-<br>C2IN0- | —                 | —                       | —                    | —                   | —                      | —                                                                                            | —                   | —                   | CLCIN0 <sup>(1)</sup> | —   |
| RA1 | 3                      | 28            | ANA1                         | —                 | —        | C1IN1-<br>C2IN1- | —                 | —                       | —                    | —                   | —                      | —                                                                                            | —                   | —                   | CLCIN1 <sup>(1)</sup> | —   |
| RA2 | 4                      | 1             | ANA2                         | VREF-             | DAC1OUT1 | C1IN0+<br>C2IN0+ | —                 | —                       | —                    | —                   | —                      | —                                                                                            | —                   | —                   | —                     | —   |
| RA3 | 5                      | 2             | ANA3                         | VREF+             | —        | C1IN1+           | —                 | —                       | —                    | —                   | MD1CARL <sup>(1)</sup> | —                                                                                            | —                   | —                   | —                     | —   |
| RA4 | 6                      | 3             | ANA4                         | —                 | —        | —                | —                 | —                       | —                    | —                   | MD1CARH <sup>(1)</sup> | T0CKI <sup>(1)</sup>                                                                         | —                   | —                   | —                     | —   |
| RA5 | 7                      | 4             | ANA5                         | —                 | —        | —                | —                 | —                       | SS1 <sup>(1,3)</sup> | —                   | MD1SRC <sup>(1)</sup>  | —                                                                                            | —                   | —                   | —                     | —   |
| RA6 | 10                     | 7             | ANA6                         | —                 | —        | —                | —                 | —                       | —                    | —                   | —                      | —                                                                                            | —                   | —                   | —                     | —   |
| RA7 | 9                      | 6             | ANA7                         | —                 | —        | —                | —                 | —                       | —                    | —                   | —                      | —                                                                                            | —                   | —                   | —                     | —   |
| RB0 | 21                     | 18            | ANB0                         | —                 | —        | C2IN1+           | ZCD               | —                       | —                    | —                   | —                      | —                                                                                            | CCP4 <sup>(1)</sup> | CWG1 <sup>(1)</sup> | —                     | —   |
| RB1 | 22                     | 19            | ANB1                         | —                 | —        | C1IN3-<br>C2IN3- | —                 | SCL2 <sup>(1,3,4)</sup> | —                    | —                   | —                      | —                                                                                            | —                   | CWG2 <sup>(1)</sup> | —                     | —   |
| RB2 | 23                     | 20            | ANB2                         | —                 | —        | —                | —                 | SDA2 <sup>(1,3,4)</sup> | —                    | —                   | —                      | —                                                                                            | —                   | CWG3 <sup>(1)</sup> | —                     | —   |
| RB3 | 24                     | 21            | ANB3                         | —                 | —        | C1IN2-<br>C2IN2- | —                 | —                       | —                    | —                   | —                      | —                                                                                            | —                   | —                   | —                     | —   |
| RB4 | 25                     | 22            | ANB4<br>ADACT <sup>(1)</sup> | —                 | —        | —                | —                 | —                       | —                    | —                   | —                      | T5G <sup>(1)</sup><br>SMT2WIN <sup>(1)</sup>                                                 | —                   | —                   | CLCIN2 <sup>(1)</sup> | —   |
| RB5 | 26                     | 23            | ANB5                         | —                 | —        | —                | —                 | —                       | —                    | —                   | —                      | T1G <sup>(1)</sup><br>SMT2SIG <sup>(1)</sup>                                                 | CCP3 <sup>(1)</sup> | —                   | CLCIN3 <sup>(1)</sup> | —   |
| RB6 | 27                     | 24            | ANB6                         | —                 | —        | —                | —                 | —                       | —                    | CTS2 <sup>(1)</sup> | —                      | —                                                                                            | —                   | —                   | —                     | —   |
| RB7 | 28                     | 25            | ANB7                         | —                 | DAC1OUT2 | —                | —                 | —                       | —                    | RX2 <sup>(1)</sup>  | —                      | T6IN <sup>(1)</sup>                                                                          | —                   | —                   | —                     | —   |
| RC0 | 11                     | 8             | ANC0                         | —                 | —        | —                | —                 | —                       | —                    | —                   | —                      | T1CKI <sup>(1)</sup><br>T3CKI <sup>(1)</sup><br>T3G <sup>(1)</sup><br>SMT1WIN <sup>(1)</sup> | —                   | —                   | —                     | —   |
| RC1 | 12                     | 9             | ANC1                         | —                 | —        | —                | —                 | —                       | —                    | —                   | —                      | SMT1SIG <sup>(1)</sup>                                                                       | CCP2 <sup>(1)</sup> | —                   | —                     | —   |
| RC2 | 13                     | 10            | ANC2                         | —                 | —        | —                | —                 | —                       | —                    | —                   | —                      | T5CKI <sup>(1)</sup>                                                                         | CCP1 <sup>(1)</sup> | —                   | —                     | —   |
| RC3 | 14                     | 11            | ANC3                         | —                 | —        | —                | —                 | SCL1 <sup>(1)</sup>     | —                    | —                   | —                      | T2IN <sup>(1)</sup>                                                                          | —                   | —                   | —                     | —   |

**Note** 1: This is a PPS remappable input signal. The input function may be moved from the default location shown to one of several other PORTx pins.

2: All output signals shown in this row are PPS remappable.

3: This is a bidirectional signal. For normal module operation, the firmware should map this signal to the same pin in both the PPS input and PPS output registers.

4: These pins can be configured for I<sup>2</sup>C and SMBTM 3.0/2.0 logic levels; the SCLx/SDAx signals may be assigned to any of the RB1/RB2/RC3/RC4 pins. PPS assignments to the other pins (e.g. RA0-RA7) will be standard TTL/ST as selected by the INLVL register, instead of the I<sup>2</sup>C specific or SMBUS input buffer thresholds.

**TABLE 3: 28-PIN ALLOCATION TABLE (PIC18(L)F25/26K83) (CONTINUED)**

| I/O                | 28-Pin SPDIF/SOIC/SSOP | 28-Pin (U)QFN | ADC              | Voltage Reference | DAC | Comparators    | Zero Cross Detect | I <sup>2</sup> C             | SPI                 | UART                                       | DSM | Timers/SMT          | CCP and PWM                                                              | CWG                                                                                                      | CLC                                      | NCO |
|--------------------|------------------------|---------------|------------------|-------------------|-----|----------------|-------------------|------------------------------|---------------------|--------------------------------------------|-----|---------------------|--------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------|------------------------------------------|-----|
| RC4                | 15                     | 12            | ANC4             | —                 | —   | —              | —                 | SDA1 <sup>(1)</sup>          | SDI1 <sup>(1)</sup> | —                                          | —   | —                   | —                                                                        | —                                                                                                        | —                                        | —   |
| RC5                | 16                     | 13            | ANC5             | —                 | —   | —              | —                 | —                            | —                   | —                                          | —   | T4IN <sup>(1)</sup> | —                                                                        | —                                                                                                        | —                                        | —   |
| RC6                | 17                     | 14            | ANC6             | —                 | —   | —              | —                 | —                            | —                   | CTS1 <sup>(1)</sup>                        | —   | —                   | —                                                                        | —                                                                                                        | —                                        | —   |
| RC7                | 18                     | 15            | ANC7             | —                 | —   | —              | —                 | —                            | —                   | RX1 <sup>(1)</sup>                         | —   | —                   | —                                                                        | —                                                                                                        | —                                        | —   |
| RE3                | 1                      | 26            | —                | —                 | —   | —              | —                 | —                            | —                   | —                                          | —   | —                   | —                                                                        | —                                                                                                        | —                                        | —   |
| VDD                | 20                     | 17            | —                | —                 | —   | —              | —                 | —                            | —                   | —                                          | —   | —                   | —                                                                        | —                                                                                                        | —                                        | —   |
| VSS                | 8, 19                  | 5, 16         | —                | —                 | —   | —              | —                 | —                            | —                   | —                                          | —   | —                   | —                                                                        | —                                                                                                        | —                                        | —   |
| OUT <sup>(2)</sup> | —                      | —             | ADGRDA<br>ADGRDB | —                 | —   | C1OUT<br>C2OUT | —                 | SDA1<br>SCL1<br>SDA2<br>SCL2 | SS1<br>SCK1<br>SDO1 | DTR1<br>RTS1<br>TX1<br>DTR2<br>RTS2<br>TX2 | DSM | TMR0                | CCP1<br>CCP2<br>CCP3<br>CCP4<br>PWM5OUT<br>PWM6OUT<br>PWM7OUT<br>PWM8OUT | CWG1A<br>CWG1B<br>CWG1C<br>CWG1D<br>CWG2A<br>CWG2B<br>CWG2C<br>CWG2D<br>CWG3A<br>CWG3B<br>CWG3C<br>CWG3D | CLC1OUT<br>CLC2OUT<br>CLC3OUT<br>CLC4OUT | NCO |

**Note** 1: This is a PPS remappable input signal. The input function may be moved from the default location shown to one of several other PORTx pins.

2: All output signals shown in this row are PPS remappable.

3: This is a bidirectional signal. For normal module operation, the firmware should map this signal to the same pin in both the PPS input and PPS output registers.

4: These pins can be configured for I<sup>2</sup>C and SMBTM 3.0/2.0 logic levels; the SCLx/SDAx signals may be assigned to any of the RB1/RB2/RC3/RC4 pins. PPS assignments to the other pins (e.g. OUT<sup>(2)</sup>) will be standard TTL/ST as selected by the INLVL register, instead of the I<sup>2</sup>C specific or SMBUS input buffer thresholds.

## Table of Contents

|      |                                                                                |     |
|------|--------------------------------------------------------------------------------|-----|
| 1.0  | Device Overview .....                                                          | 10  |
| 2.0  | Guidelines for Getting Started with PIC18(L)F25/26K83 Microcontrollers .....   | 13  |
| 3.0  | PIC18 CPU .....                                                                | 16  |
| 4.0  | Memory Organization .....                                                      | 23  |
| 5.0  | Device Configuration .....                                                     | 55  |
| 6.0  | Resets .....                                                                   | 71  |
| 7.0  | Oscillator Module (with Fail-Safe Clock Monitor) .....                         | 82  |
| 8.0  | Reference Clock Output Module .....                                            | 101 |
| 9.0  | Interrupt Controller .....                                                     | 105 |
| 10.0 | Power-Saving Operation Modes .....                                             | 161 |
| 11.0 | Windowed Watchdog Timer (WWDT) .....                                           | 168 |
| 12.0 | 8x8 Hardware Multiplier .....                                                  | 177 |
| 13.0 | Nonvolatile Memory (NVM) Control .....                                         | 179 |
| 14.0 | Cyclic Redundancy Check (CRC) Module with Memory Scanner .....                 | 203 |
| 15.0 | Direct Memory Access (DMA) .....                                               | 218 |
| 16.0 | I/O Ports .....                                                                | 250 |
| 17.0 | Peripheral Pin Select (PPS) Module .....                                       | 263 |
| 18.0 | Interrupt-on-Change .....                                                      | 271 |
| 19.0 | Peripheral Module Disable (PMD) .....                                          | 275 |
| 20.0 | Timer0 Module .....                                                            | 284 |
| 21.0 | Timer1/3/5 Module with Gate Control .....                                      | 290 |
| 22.0 | Timer2/4/6 Module .....                                                        | 305 |
| 23.0 | Capture/Compare/PWM Module .....                                               | 327 |
| 24.0 | Pulse-Width Modulation (PWM) .....                                             | 341 |
| 25.0 | Signal Measurement Timer (SMTX) .....                                          | 348 |
| 26.0 | Complementary Waveform Generator (CWG) Module .....                            | 392 |
| 27.0 | Configurable Logic Cell (CLC) .....                                            | 420 |
| 28.0 | Numerically Controlled Oscillator (NCO) Module .....                           | 435 |
| 29.0 | Zero-Cross Detection (ZCD) Module .....                                        | 445 |
| 30.0 | Data Signal Modulator (DSM) Module .....                                       | 450 |
| 31.0 | Universal Asynchronous Receiver Transmitter (UART) With Protocol Support ..... | 461 |
| 32.0 | Serial Peripheral Interface (SPI) Module .....                                 | 498 |
| 33.0 | I2C Module .....                                                               | 530 |
| 34.0 | CAN Module .....                                                               | 583 |
| 35.0 | Fixed Voltage Reference (FVR) .....                                            | 650 |
| 36.0 | Temperature Indicator Module .....                                             | 652 |
| 37.0 | Analog-to-Digital Converter with Computation (ADC2) Module .....               | 654 |
| 38.0 | 5-Bit Digital-to-Analog Converter (DAC) Module .....                           | 692 |
| 39.0 | Comparator Module .....                                                        | 696 |
| 40.0 | High/Low-Voltage Detect (HLVD) .....                                           | 705 |
| 41.0 | In-Circuit Serial Programming™ (ICSP™) .....                                   | 713 |
| 42.0 | Instruction Set Summary .....                                                  | 715 |
| 43.0 | Register Summary .....                                                         | 769 |
| 44.0 | Development Support .....                                                      | 790 |
| 45.0 | Electrical Specifications .....                                                | 794 |
| 46.0 | DC and AC Characteristics Graphs and Tables .....                              | 825 |
| 47.0 | Packaging Information .....                                                    | 826 |
|      | The Microchip Website .....                                                    | 841 |
|      | Customer Change Notification Service .....                                     | 841 |
|      | Customer Support .....                                                         | 841 |
|      | Product Identification System .....                                            | 842 |



## TO OUR VALUED CUSTOMERS

It is our intention to provide our valued customers with the best documentation possible to ensure successful use of your Microchip products. To this end, we will continue to improve our publications to better suit your needs. Our publications will be refined and enhanced as new volumes and updates are introduced.

If you have any questions or comments regarding this publication, please contact the Marketing Communications Department via E-mail at [docerrors@microchip.com](mailto:docerrors@microchip.com). We welcome your feedback.

### Most Current Data Sheet

To obtain the most up-to-date version of this data sheet, please register at our Worldwide Website at:

<http://www.microchip.com>

You can determine the version of a data sheet by examining its literature number found on the bottom outside corner of any page. The last character of the literature number is the version number, (e.g., DS30000000A is version A of document DS30000000).

### Errata

An errata sheet, describing minor operational differences from the data sheet and recommended workarounds, may exist for current devices. As device/documentation issues become known to us, we will publish an errata sheet. The errata will specify the revision of silicon and revision of document to which it applies.

To determine if an errata sheet exists for a particular device, please check with one of the following:

- Microchip's Worldwide Website; <http://www.microchip.com>
- Your local Microchip sales office (see last page)

When contacting a sales office, please specify which device, revision of silicon and data sheet (include literature number) you are using.

### Customer Notification System

Register on our website at [www.microchip.com](http://www.microchip.com) to receive the most current information on all of our products.

## 1.0 DEVICE OVERVIEW

This document contains device specific information for the following devices:

- PIC18F25K83
- PIC18LF25K83
- PIC18F26K83
- PIC18LF26K83

This family offers the advantages of all PIC18 microcontrollers – namely, high computational performance at an economical price – with the addition of high-endurance Program Flash Memory, Universal Asynchronous Receiver Transmitter (UART), Serial Peripheral Interface (SPI), Inter-integrated Circuit (I<sup>2</sup>C), Direct Memory Access (DMA), Configurable Logic Cells (CLC), Signal Measurement Timer (SMT), Numerically Controlled Oscillator (NCO), and Analog-to-Digital Converter with Computation (ADC<sup>2</sup>).

### 1.1 New Features

- **Direct Memory Access Controller:** The Direct Memory Access (DMA) Controller is designed to service data transfers between different memory regions directly without intervention from the CPU. By eliminating the need for CPU-intensive management of handling interrupts intended for data transfers, the CPU now can spend more time on other tasks.
- **Vectored Interrupt Controller:** The Vectored Interrupt Controller module reduces the numerous peripheral interrupt request signals to a single interrupt request signal to the CPU. It assembles all of the interrupt request signals and resolves the interrupts based on both a fixed natural order priority and a user-assigned priority, thereby eliminating scanning of interrupt sources.
- **Universal Asynchronous Receiver Transmitter:** The Universal Asynchronous Receiver Transmitter (UART) module is a serial I/O communications peripheral. It contains all the clock generators, shift registers and data buffers necessary to perform an input or output serial data transfer, independent of device program execution. The UART can be configured as a full-duplex asynchronous system or one of several automated protocols. Full-Duplex mode is useful for communications with peripheral systems, with DMX/DALI/LIN support.

- **Serial Peripheral Interface:** The Serial Peripheral Interface (SPI) module is a synchronous serial data communication bus that operates in Full-Duplex mode. Devices communicate in a master/slave environment where the master device initiates the communication. A slave device is controlled through a Chip Select known as Slave Select. Example slave devices include serial EEPROMs, shift registers, display drivers, A/D converters, or another PIC<sup>®</sup> device.
- **I<sup>2</sup>C Module:** The I<sup>2</sup>C module provides a synchronous interface between the microcontroller and other I<sup>2</sup>C-compatible devices using the two-wire I<sup>2</sup>C serial bus. Devices communicate in a master/slave environment. The I<sup>2</sup>C bus specifies two signal connections – Serial Clock (SCL) and Serial Data (SDA). Both the SCL and SDA connections are bidirectional open-drain lines, each requiring pull-up resistors to the supply voltage.
- **12-bit A/D Converter with Computation:** This module incorporates programmable acquisition time, allowing for a channel to be selected and a conversion to be initiated without waiting for a sampling period and thus, reduce code overhead. It has a new module called ADC<sup>2</sup> with computation features, which provides a digital filter and threshold interrupt functions.

### 1.2 Details on Individual Family Members

Devices in the PIC18(L)F25/26K83 family are available in 28-pin packages. The block diagram for this device is shown in [Figure 3-1](#).

The similarities and differences among the devices are listed in the PIC18(L)F25/26K83 Family Types Table (page 4). The pinouts for all devices are listed in [Table 3](#).

# PIC18(L)F25/26K83

**TABLE 1-1: DEVICE FEATURES**

| Features                                                                                    | PIC18(L)F25K83                                                                                                        | PIC18(L)F26K83                                                          |
|---------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------|
| Program Memory (Bytes)                                                                      | 32768                                                                                                                 | 65536                                                                   |
| Program Memory (Instructions)                                                               | 16384                                                                                                                 | 32768                                                                   |
| Data Memory (Bytes)                                                                         | 2048                                                                                                                  | 4096                                                                    |
| Data EEPROM Memory (Bytes)                                                                  | 1024                                                                                                                  | 1024                                                                    |
| Packages                                                                                    | 28-pin SPDIP<br>28-pin SOIC<br>28-pin SSOP<br>28-pin QFN<br>28-pin UQFN                                               | 28-pin SPDIP<br>28-pin SOIC<br>28-pin SSOP<br>28-pin QFN<br>28-pin UQFN |
| I/O Ports                                                                                   | A,B,C,E <sup>(1)</sup>                                                                                                | A,B,C,E <sup>(1)</sup>                                                  |
| 12-Bit Analog-to-Digital Conversion Module (ADC <sup>2</sup> ) with Computation Accelerator | 5 internal<br>24 external                                                                                             | 5 internal<br>24 external                                               |
| Capture/Compare/PWM Modules (CCP)                                                           | 4                                                                                                                     |                                                                         |
| 10-Bit Pulse-Width Modulator (PWM)                                                          | 4                                                                                                                     |                                                                         |
| Timers (16-/8-bit)                                                                          | 4/3                                                                                                                   |                                                                         |
| Serial Communications                                                                       | 2 UARTs with DMX/DALI/LIN, 2 I <sup>2</sup> C, 1 SPI                                                                  |                                                                         |
| Complementary Waveform Generator (CWG)                                                      | 3                                                                                                                     |                                                                         |
| Zero-Cross Detect (ZCD)                                                                     | 1                                                                                                                     |                                                                         |
| Data Signal Modulator (DSM)                                                                 | 1                                                                                                                     |                                                                         |
| Signal Measurement Timer (SMT)                                                              | 2                                                                                                                     |                                                                         |
| 5-bit Digital to Analog Converter (DAC)                                                     | 1                                                                                                                     |                                                                         |
| Numerically Controlled Oscillator (NCO)                                                     | 1                                                                                                                     |                                                                         |
| Comparator Module                                                                           | 2                                                                                                                     |                                                                         |
| Direct Memory Access (DMA)                                                                  | 2                                                                                                                     |                                                                         |
| Configurable Logic Cell (CLC)                                                               | 4                                                                                                                     |                                                                         |
| Control Area Network (CAN)                                                                  | Yes                                                                                                                   |                                                                         |
| Peripheral Module Disable (PMD)                                                             | Yes                                                                                                                   |                                                                         |
| 16-bit CRC with Scanner                                                                     | Yes                                                                                                                   |                                                                         |
| Programmable High/Low-Voltage Detect (HLVD)                                                 | Yes                                                                                                                   |                                                                         |
| Resets (and Delays)                                                                         | POR, Programmable BOR,<br>RESET Instruction,<br>Stack Overflow,<br>Stack Underflow<br>(PWRT, OST),<br>MCLR, WDT, MEMV |                                                                         |
| Instruction Set                                                                             | 81 Instructions;<br>87 with Extended Instruction Set enabled                                                          |                                                                         |
| Maximum Operating Frequency                                                                 | 64 MHz                                                                                                                |                                                                         |

**Note 1:** PORTE contains the single RE3 input-only pin.

## 1.3 Register and Bit naming conventions

### 1.3.1 REGISTER NAMES

When there are multiple instances of the same peripheral in a device, the peripheral control registers will be depicted as the concatenation of a peripheral identifier, peripheral instance, and control identifier. The control registers section will show just one instance of all the register names with an 'x' in the place of the peripheral instance number. This naming convention may also be applied to peripherals when there is only one instance of that peripheral in the device to maintain compatibility with other devices in the family that contain more than one.

### 1.3.2 BIT NAMES

There are two variants for bit names:

- Short name: Bit function abbreviation
- Long name: Peripheral abbreviation + short name

#### 1.3.2.1 Short Bit Names

Short bit names are an abbreviation for the bit function. For example, some peripherals are enabled with the EN bit. The bit names shown in the registers are the short name variant.

Short bit names are useful when accessing bits in C programs. The general format for accessing bits by the short name is *RegisterNamebits.ShortName*. For example, the enable bit, EN, in the T0CON0 register can be set in C programs with the instruction `T0CON0bits.EN = 1`.

Short names are generally not useful in assembly programs because the same name may be used by different peripherals in different bit positions. When this occurs, during the include file generation, all instances of that short bit name are appended with an underscore plus the name of the register in which the bit resides to avoid naming contentions.

#### 1.3.2.2 Long Bit Names

Long bit names are constructed by adding a peripheral abbreviation prefix to the short name. The prefix is unique to the peripheral thereby making every long bit name unique. The long bit name for the Timer0 enable bit is the Timer0 prefix, T0, appended with the enable bit short name, EN, resulting in the unique bit name T0EN.

Long bit names are useful in both C and assembly programs. For example, in C the T0CON0 enable bit can be set with the `T0EN = 1` instruction. In assembly, this bit can be set with the `BSF T0CON0, T0EN` instruction.

#### 1.3.2.3 Bit Fields

Bit fields are two or more adjacent bits in the same register. For example, the four Least Significant bits of the T0CON0 register contain the output prescaler select bits. The short name for this field is OUTPS and the long name is T0OUTPS. Bit field access is only possible in C programs. The following example demonstrates a C program instruction for setting the Timer0 output prescaler to the 1:6 Postscaler:

```
T0CON0bits.OUTPS = 0x5;
```

Individual bits in a bit field can also be accessed with long and short bit names. Each bit is the field name appended with the number of the bit position within the field. For example, the Most Significant mode bit has the short bit name OUTPS3. The following two examples demonstrate assembly program sequences for setting the Timer0 output prescaler to 1:6 Postscaler:

Example 1:

```
MOVLW  ~(1<<OUTPS3 | 1<<OUTPS1)
ANDWF  T0CON0,F
MOVLW  1<<OUTPS2 | 1<<OUTPS0
IORWF  T0CON0,F
```

Example 2:

```
BCF    T0CON0,OUTPS3
BSF    T0CON0,OUTPS2
BCF    T0CON0,OUTPS1
BSF    T0CON0,OUTPS0
```

## 1.3.3 REGISTER AND BIT NAMING EXCEPTIONS

### 1.3.3.1 Status, Interrupt, and Mirror Bits

Status, interrupt enables, interrupt flags, and mirror bits are contained in registers that span more than one peripheral. In these cases, the bit name shown is unique so there is no prefix or short name variant.

## 2.0 GUIDELINES FOR GETTING STARTED WITH PIC18(L)F25/26K83 MICROCONTROLLERS

### 2.1 Basic Connection Requirements

Getting started with the PIC18(L)F25/26K83 family of 8-bit microcontrollers requires attention to a minimal set of device pin connections before proceeding with development.

The following pins must always be connected:

- All VDD and VSS pins (see [Section 2.2 “Power Supply Pins”](#))
- MCLR pin (see [Section 2.3 “Master Clear \(MCLR\) Pin”](#))

These pins must also be connected if they are being used in the end application:

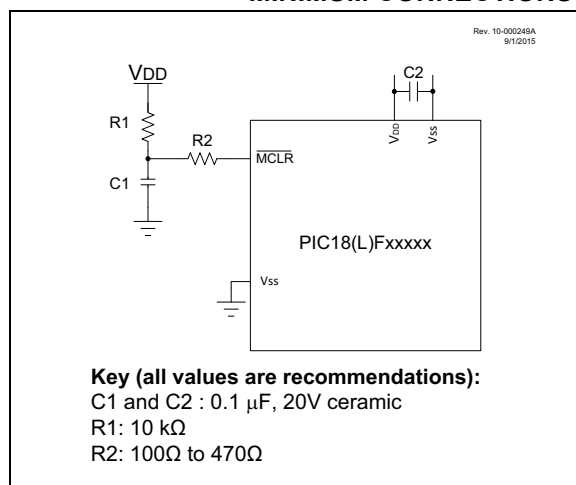
- ICSPCLK/ICSPDAT pins used for In-Circuit Serial Programming™ (ICSP™) and debugging purposes (see [Section 2.4 “ICSP™ Pins”](#))
- OSCI and OSCO pins when an external oscillator source is used (see [Section 2.5 “External Oscillator Pins”](#))

Additionally, the following pins may be required:

- VREF+/VREF- pins are used when external voltage reference for analog modules is implemented

The minimum mandatory connections are shown in [Figure 2-1](#).

**FIGURE 2-1: RECOMMENDED MINIMUM CONNECTIONS**



### 2.2 Power Supply Pins

#### 2.2.1 DECOUPLING CAPACITORS

The use of decoupling capacitors on every pair of power supply pins (VDD and VSS) is required.

Consider the following criteria when using decoupling capacitors:

- **Value and type of capacitor:** A 0.1  $\mu$ F (100 nF), 10-20V capacitor is recommended. The capacitor should be a low-ESR device, with a resonance frequency in the range of 200 MHz and higher. Ceramic capacitors are recommended.
- **Placement on the printed circuit board:** The decoupling capacitors should be placed as close to the pins as possible. It is recommended to place the capacitors on the same side of the board as the device. If space is constricted, the capacitor can be placed on another layer on the PCB using a via; however, ensure that the trace length from the pin to the capacitor is no greater than 0.25 inch (6 mm).
- **Handling high-frequency noise:** If the board is experiencing high-frequency noise (upward of tens of MHz), add a second ceramic type capacitor in parallel to the above described decoupling capacitor. The value of the second capacitor can be in the range of 0.01  $\mu$ F to 0.001  $\mu$ F. Place this second capacitor next to each primary decoupling capacitor. In high-speed circuit designs, consider implementing a decade pair of capacitances as close to the power and ground pins as possible (e.g., 0.1  $\mu$ F in parallel with 0.001  $\mu$ F).
- **Maximizing performance:** On the board layout from the power supply circuit, run the power and return traces to the decoupling capacitors first, and then to the device pins. This ensures that the decoupling capacitors are first in the power chain. Equally important is to keep the trace length between the capacitor and the power pins to a minimum, thereby reducing PCB trace inductance.

#### 2.2.2 TANK CAPACITORS

On boards with power traces running longer than six inches in length, it is suggested to use a tank capacitor for integrated circuits, including microcontrollers, to supply a local power source. The value of the tank capacitor should be determined based on the trace resistance that connects the power supply source to the device, and the maximum current drawn by the device in the application. In other words, select the tank capacitor so that it meets the acceptable voltage sag at the device. Typical values range from 4.7  $\mu$ F to 47  $\mu$ F.

## 2.3 Master Clear ( $\overline{\text{MCLR}}$ ) Pin

The  $\overline{\text{MCLR}}$  pin provides two specific device functions: Device Reset, and Device Programming and Debugging. If programming and debugging are not required in the end application, a direct connection to VDD may be all that is required. The addition of other components, to help increase the application's resistance to spurious Resets from voltage sags, may be beneficial. A typical configuration is shown in Figure 2-1. Other circuit designs may be implemented, depending on the application requirements.

During programming and debugging, the resistance and capacitance that can be added to the pin must be considered. Device programmers and debuggers drive the  $\overline{\text{MCLR}}$  pin. Consequently, specific voltage levels ( $V_{IH}$  and  $V_{IL}$ ) and fast signal transitions must not be adversely affected. Therefore, specific values of R1 and C1 will need to be adjusted based on the application and PCB requirements. For example, it is recommended that the capacitor, C1, be isolated from the  $\overline{\text{MCLR}}$  pin during programming and debugging operations by using a jumper (Figure 2-2). The jumper is replaced for normal run-time operations.

Any components associated with the  $\overline{\text{MCLR}}$  pin should be placed within 0.25 inch (6 mm) of the pin.

## 2.4 ICSP™ Pins

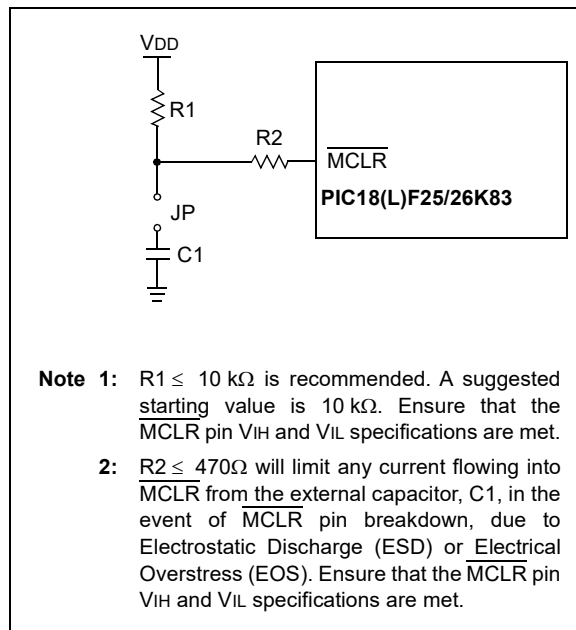
The ICSPCLK and ICSPDAT pins are used for In-Circuit Serial Programming™ (ICSP™) and debugging purposes. It is recommended to keep the trace length between the ICSP connector and the ICSP pins on the device as short as possible. If the ICSP connector is expected to experience an ESD event, a series resistor is recommended, with the value in the range of a few tens of ohms, not to exceed 100Ω.

Pull-up resistors, series diodes and capacitors on the ICSPCLK and ICSPDAT pins are not recommended as they will interfere with the programmer/debugger communications to the device. If such discrete components are an application requirement, they should be removed from the circuit during programming and debugging. Alternatively, refer to the AC/DC characteristics and timing requirements information in the respective device Flash programming specification for information on capacitive loading limits, and pin input voltage high ( $V_{IH}$ ) and input low ( $V_{IL}$ ) requirements.

For device emulation, ensure that the “Communication Channel Select” (i.e., ICSPCLK/ICSPDAT pins), programmed into the device, matches the physical connections for the ICSP to the Microchip debugger/emulator tool.

For more information on available Microchip development tools connection requirements, refer to [Section 44.0 “Development Support”](#).

**FIGURE 2-2: EXAMPLE OF  $\overline{\text{MCLR}}$  PIN CONNECTIONS**



## 2.5 External Oscillator Pins

Many microcontrollers have options for at least two oscillators: a high-frequency primary oscillator and a low-frequency secondary oscillator (refer to [Section 7.0 “Oscillator Module \(with Fail-Safe Clock Monitor\)”](#) for details).

The oscillator circuit should be placed on the same side of the board as the device. Place the oscillator circuit close to the respective oscillator pins with no more than 0.5 inch (12 mm) between the circuit components and the pins. The load capacitors should be placed next to the oscillator itself, on the same side of the board.

Use a grounded copper pour around the oscillator circuit to isolate it from surrounding circuits. The grounded copper pour should be routed directly to the MCU ground. Do not run any signal traces or power traces inside the ground pour. Also, if using a two-sided board, avoid any traces on the other side of the board where the crystal is placed.

Layout suggestions are shown in [Figure 2-3](#). In-line packages may be handled with a single-sided layout that completely encompasses the oscillator pins. With fine-pitch packages, it is not always possible to completely surround the pins and components. A suitable solution is to tie the broken guard sections to a mirrored ground layer. In all cases, the guard trace(s) must be returned to ground.

In planning the application's routing and I/O assignments, ensure that adjacent port pins, and other signals in close proximity to the oscillator, are benign (i.e., free of high frequencies, short rise and fall times, and other similar noise).

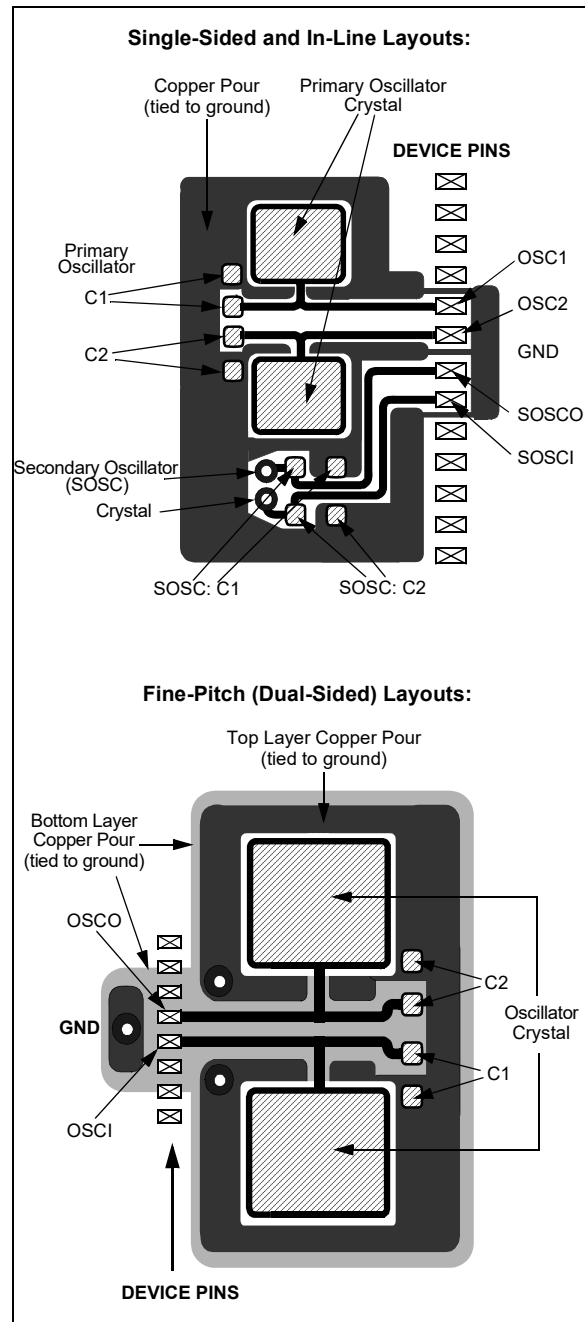
For additional information and design guidance on oscillator circuits, refer to these Microchip Application Notes, available at the corporate website ([www.microchip.com](http://www.microchip.com)):

- AN826, “Crystal Oscillator Basics and Crystal Selection for *rPIC™* and *PICmicro®* Devices”
- AN849, “Basic *PICmicro®* Oscillator Design”
- AN943, “Practical *PICmicro®* Oscillator Analysis and Design”
- AN949, “Making Your Oscillator Work”

## 2.6 Unused I/Os

Unused I/O pins should be configured as outputs and driven to a logic low state. Alternatively, connect a 1 kΩ to 10 kΩ resistor to Vss on unused pins and drive the output to logic low.

**FIGURE 2-3: SUGGESTED PLACEMENT OF THE OSCILLATOR CIRCUIT**



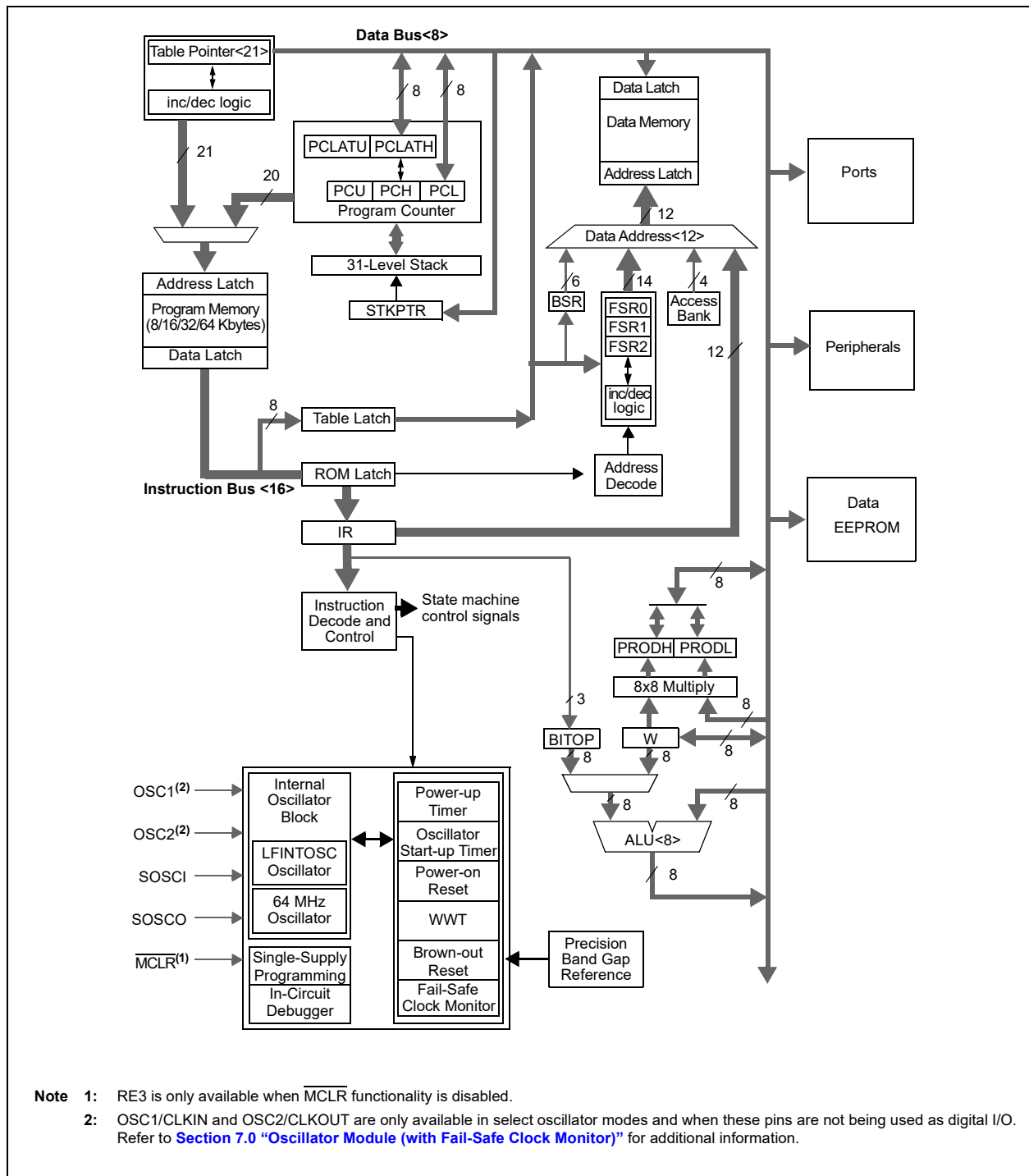
## 3.0 PIC18 CPU

This family of devices contains a PIC18 8-bit CPU core based on the modified Harvard architecture. The PIC18 CPU supports:

- System Arbitration which decides memory access allocation depending on user priorities
- Vectored Interrupt capability with automatic two level deep context saving
- 31-level deep hardware stack with overflow and underflow reset capabilities
- Support Direct, Indirect, and Relative Addressing modes
- 8x8 Hardware Multiplier



**FIGURE 3-1: PIC18(L)F25/26K83 FAMILY BLOCK DIAGRAM**



## 3.1 System Arbitration

The System Arbiter resolves memory access between the System Level Selections (i.e., Main, Interrupt Service Routine) and Peripheral Selection (i.e., DMA and Scanner) based on user-assigned priorities. Each of the system level and peripheral selections has its own priority selection registers. Memory access priority is resolved using the number written to the corresponding Priority registers, 0 being the highest priority and 4 the lowest. The default priorities are listed in [Table 3-1](#).

In case the user wants to change priorities, ensure each Priority register is written with a unique value from 0 to 4.

**TABLE 3-1: DEFAULT PRIORITIES**

| Selection    |         | Priority register<br>Reset value |
|--------------|---------|----------------------------------|
| System Level | ISR     | 0                                |
|              | MAIN    | 1                                |
| Peripheral   | DMA1    | 2                                |
|              | DMA2    | 3                                |
|              | SCANNER | 4                                |

## 3.1.1 PRIORITY LOCK

The System arbiter grants memory access to the peripheral selections (DMAx, Scanner) when the PRLOCKED bit (PRLOCK Register) is set.

Priority selections are locked by setting the PRLOCKED bit of the PRLOCK register. Setting and clearing this bit requires a special sequence as an extra precaution against inadvertent changes. Examples of setting and clearing the PRLOCKED bit are shown in [Example 3-1](#) and [Example 3-2](#).

### EXAMPLE 3-1: PRIORITY LOCK SEQUENCE

```
; Disable interrupts
BCF INTCON0,GIE

; Bank to PRLOCK register
BANKSEL PRLOCK
MOVLW 55h

; Required sequence, next 4
instructions
MOVWF PRLOCK
MOVLW AAh
MOVWF PRLOCK
; Set PRLOCKED bit to grant memory
access to peripherals
BSF PRLOCK,0

; Enable Interrupts
BSF INTCON0,GIE
```

### EXAMPLE 3-2: PRIORITY UNLOCK SEQUENCE

```
; Disable interrupts
BCF INTCON0,GIE

; Bank to PRLOCK register
BANKSEL PRLOCK
MOVLW 55h

; Required sequence, next 4
instructions
MOVWF PRLOCK
MOVLW AAh
MOVWF PRLOCK
; Clear PRLOCKED bit to allow changing
priority settings
BCF PRLOCK,0

; Enable Interrupts
BSF INTCON0,GIE
```

## 3.2 Memory Access Scheme

The user can assign priorities to both system level and peripheral selections based on which the system arbiter grants memory access. Let us consider the following priority scenarios between ISR, MAIN, and Peripherals.

**Note:** It is always required that the ISR priority be higher than Main priority.

### 3.2.1 ISR PRIORITY > MAIN PRIORITY > PERIPHERAL PRIORITY

When the Peripheral Priority (DMAx, Scanner) is lower than ISR and MAIN Priority, and the peripheral requires:

1. Access to the Program Flash Memory, then the peripheral waits for an instruction cycle in which the CPU does not need to access the PFM (such as a branch instruction) and uses that cycle to do its own Program Flash Memory access, unless a PFM Read/Write operation is in progress.
2. Access to the SFR/GPR, then the peripheral waits for an instruction cycle in which the CPU does not need to access the SFR/GPR (such as MOVLW, CALL, NOP) and uses that cycle to do its own SFR/GPR access.
3. Access to the Data EEPROM, then the peripheral has access to Data EEPROM unless a Data EEPROM Read/Write operation is being performed.

This results in the lowest throughput for the peripheral to access the memory, and does so without any impact on execution times.

### 3.2.2 PERIPHERAL PRIORITY > ISR PRIORITY > MAIN PRIORITY

When the Peripheral Priority (DMAx, Scanner) is higher than ISR and MAIN Priority, the CPU operation is stalled when the peripheral requests memory.

The CPU is held in its current state until the peripheral completes its operation. Since the peripheral requests access to the bus, the peripheral cannot be disabled until it completes its operation.

This results in the highest throughput for the peripheral to access the memory, but has the cost of stalling other execution while it occurs.

## 3.2.3 ISR PRIORITY > PERIPHERAL PRIORITY > MAIN PRIORITY

In this case, interrupt routines and peripheral operation (DMAx, Scanner) will stall the CPU. Interrupt will preempt peripheral operation. This results in lowest interrupt latency and highest throughput for the peripheral to access the memory.

## 3.2.4 PERIPHERAL 1 PRIORITY > ISR PRIORITY > MAIN PRIORITY > PERIPHERAL 2 PRIORITY

In this case, the Peripheral 1 will stall the execution of the CPU. However, Peripheral 2 can access the memory in cycles unused by Peripheral 1.

The operation of the System Arbiter is controlled through the following registers:

### REGISTER 3-1: ISRPR: INTERRUPT SERVICE ROUTINE PRIORITY REGISTER

|       |     |     |     |     |            |         |         |
|-------|-----|-----|-----|-----|------------|---------|---------|
| U-0   | U-0 | U-0 | U-0 | U-0 | R/W-0/0    | R/W-0/0 | R/W-0/0 |
| —     | —   | —   | —   | —   | ISRPR<2:0> |         |         |
| bit 7 |     |     |     |     | bit 0      |         |         |

#### Legend:

|                      |                    |                                                       |
|----------------------|--------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit   | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| 1 = bit is set       | 0 = bit is cleared | HS = Hardware set                                     |

bit 7-3 **Unimplemented:** Read as '0'

bit 2-0 **ISRPR<2:0>**: Interrupt Service Routine Priority Selection bits

### REGISTER 3-2: MAINPR: MAIN ROUTINE PRIORITY REGISTER

|       |     |     |     |     |             |         |         |
|-------|-----|-----|-----|-----|-------------|---------|---------|
| U-0   | U-0 | U-0 | U-0 | U-0 | R/W-0/0     | R/W-0/0 | R/W-1/1 |
| —     | —   | —   | —   | —   | MAINPR<2:0> |         |         |
| bit 7 |     |     |     |     | bit 0       |         |         |

#### Legend:

|                      |                    |                                                       |
|----------------------|--------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit   | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| 1 = bit is set       | 0 = bit is cleared | HS = Hardware set                                     |

bit 7-3 **Unimplemented:** Read as '0'

bit 2-0 **MAINPR<2:0>**: Main Routine Priority Selection bits

### REGISTER 3-3: DMA1PR: DMA1 PRIORITY REGISTER

|       |     |     |     |     |             |         |         |
|-------|-----|-----|-----|-----|-------------|---------|---------|
| U-0   | U-0 | U-0 | U-0 | U-0 | R/W-0/0     | R/W-1/1 | R/W-0/0 |
| —     | —   | —   | —   | —   | DMA1PR<2:0> |         |         |
| bit 7 |     |     |     |     | bit 0       |         |         |

#### Legend:

|                      |                    |                                                       |
|----------------------|--------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit   | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| 1 = bit is set       | 0 = bit is cleared | HS = Hardware set                                     |

bit 7-3 **Unimplemented:** Read as '0'

bit 2-0 **DMA1PR<2:0>**: DMA1 Priority Selection bits

## REGISTER 3-4: DMA2PR: DMA2 PRIORITY REGISTER

|       |     |     |     |     |             |         |         |
|-------|-----|-----|-----|-----|-------------|---------|---------|
| U-0   | U-0 | U-0 | U-0 | U-0 | R/W-0/0     | R/W-1/1 | R/W-1/1 |
| —     | —   | —   | —   | —   | DMA2PR<2:0> |         |         |
| bit 7 |     |     |     |     | bit 0       |         |         |

### Legend:

R = Readable bit      W = Writable bit      U = Unimplemented bit, read as '0'  
 u = Bit is unchanged      x = Bit is unknown      -n/n = Value at POR and BOR/Value at all other Resets  
 1 = bit is set      0 = bit is cleared      HS = Hardware set

bit 7-3      **Unimplemented:** Read as '0'

bit 2-0      **DMA2PR<2:0>**: DMA2 Priority Selection bits

## REGISTER 3-5: SCANPR: SCANNER PRIORITY REGISTER

|       |     |     |     |     |             |         |         |
|-------|-----|-----|-----|-----|-------------|---------|---------|
| U-0   | U-0 | U-0 | U-0 | U-0 | R/W-1/1     | R/W-0/0 | R/W-0/0 |
| —     | —   | —   | —   | —   | SCANPR<2:0> |         |         |
| bit 7 |     |     |     |     | bit 0       |         |         |

### Legend:

R = Readable bit      W = Writable bit      U = Unimplemented bit, read as '0'  
 u = Bit is unchanged      x = Bit is unknown      -n/n = Value at POR and BOR/Value at all other Resets  
 1 = bit is set      0 = bit is cleared      HS = Hardware set

bit 7-3      **Unimplemented:** Read as '0'

bit 2-0      **SCANPR<2:0>**: Scanner Priority Selection bits

## REGISTER 3-6: PRLOCK: PRIORITY LOCK REGISTER

|       |     |     |     |     |       |     |          |
|-------|-----|-----|-----|-----|-------|-----|----------|
| U-0   | U-0 | U-0 | U-0 | U-0 | U-0   | U-0 | R/W-0/0  |
| —     | —   | —   | —   | —   | —     | —   | PRLOCKED |
| bit 7 |     |     |     |     | bit 0 |     |          |

### Legend:

R = Readable bit      W = Writable bit      U = Unimplemented bit, read as '0'  
 u = Bit is unchanged      x = Bit is unknown      -n/n = Value at POR and BOR/Value at all other Resets  
 1 = bit is set      0 = bit is cleared      HS = Hardware set

bit 7-1      **Unimplemented:** Read as '0'

bit 0      **PRLOCKED**: PR Register Lock bit<sup>(1, 2)</sup>

0 = Priority Registers can be modified by write operations; Peripherals do not have access to the memory

1 = Priority Registers are locked and cannot be written; Peripherals do not have access to the memory

**Note 1:** The PRLOCKED bit can only be set or cleared after the unlock sequence.

**2:** If PR1WAY = 1, the PRLOCKED bit cannot be cleared after it has been set. A system Reset will clear the bit and allow one more set.

**TABLE 3-2: SUMMARY OF REGISTERS ASSOCIATED WITH CPU**

| Name   | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2   | Bit 1   | Bit 0    | Register on page   |
|--------|-------|-------|-------|-------|-------|---------|---------|----------|--------------------|
| ISRPR  | —     | —     | —     | —     | —     | ISRPR2  | ISRPR1  | ISRPR0   | <a href="#">20</a> |
| MAINPR | —     | —     | —     | —     | —     | MAINPR2 | MAINPR1 | MAINPR0  | <a href="#">20</a> |
| DMA1PR | —     | —     | —     | —     | —     | DMA1PR2 | DMA1PR1 | DMA1PR0  | <a href="#">20</a> |
| DMA2PR | —     | —     | —     | —     | —     | DMA2PR2 | DMA2PR1 | DMA2PR0  | <a href="#">21</a> |
| SCANPR | —     | —     | —     | —     | —     | SCANPR2 | SCANPR1 | SCANPR0  | <a href="#">21</a> |
| PRLOCK | —     | —     | —     | —     | —     | —       | —       | PRLOCKED | <a href="#">21</a> |

**Legend:** — = Unimplemented location, read as '0'.

## 4.0 MEMORY ORGANIZATION

There are three types of memory in PIC18 enhanced microcontroller devices:

- Program Flash Memory
- Data RAM
- Data EEPROM

The Program Memory Flash and data RAM share the same bus, while data EEPROM uses a separate bus. This allows for concurrent access of the memory spaces.

Additional detailed information on the operation of the Program Flash Memory and Data EEPROM Memory is provided in [Section 13.0 “Nonvolatile Memory \(NVM\) Control”](#).

### 4.1 Program Flash Memory Organization

PIC18 microcontrollers implement a 21-bit Program Counter, which is capable of addressing a 2 Mbyte program memory space. Accessing any unimplemented memory will return all '0's (a NOP instruction).

These devices contains the following:

- PIC18(L)F25K83: 32 Kbytes of Flash memory, up to 16,384 single-word instructions
- PIC18(L)F26K83: 64 Kbytes of Flash memory, up to 32,768 single-word instructions

The Reset vector for the device is at address 000000h. PIC18(L)F25/26K83 devices feature a vectored interrupt controller with a dedicated interrupt vector table in the program memory, see [Section 9.0 “Interrupt Controller”](#).

**Note:** For memory information on this family of devices, see [Table 4-1](#) and [Table 4-3](#).

## 4.2 Memory Access Partition (MAP)

Program Flash Memory is partitioned into:

- Application Block
- Boot Block, and
- Storage Area Flash (SAF) Block

### 4.2.1 APPLICATION BLOCK

Application Block is where the user's program resides by default. Default settings of the Configuration bits ( $\overline{\text{BBEN}} = 1$  and  $\overline{\text{SAFEN}} = 1$ ) assign all memory in the Program Flash Memory area to the Application Block. The  $\overline{\text{WRTAPP}}$  Configuration bit is used to protect the Application Block.

### 4.2.2 BOOT BLOCK

Boot Block is an area in program memory that is ideal for storing bootloader code. Code placed in this area can be executed by the CPU. The Boot Block can be write-protected, independent of the main Application Block. The Boot Block is enabled by the  $\overline{\text{BBEN}}$  bit and size is based on the value of the BBSIZE bits of Configuration word ([Register 5-7](#)), see [Table 5-1](#) for Boot Block sizes.

The  $\overline{\text{WRTB}}$  Configuration bit is used to write-protect the Boot Block.

### 4.2.3 STORAGE AREA FLASH

Storage Area Flash (SAF) is the area in program memory that can be used as data storage. SAF is enabled by the  $\overline{\text{SAFEN}}$  bit of the Configuration word in [Register 5-7](#). If enabled, the code placed in this area cannot be executed by the CPU. The SAF block is placed at the end of memory and spans 128 words. The  $\overline{\text{WRTSAF}}$  Configuration bit is used to write-protect the Storage Area Flash.

**Note:** If write-protected locations are written from NVMCON registers, memory is not changed and the WRERR bit defined in [Register 13-1](#) is set.

**TABLE 4-1: PROGRAM AND DATA EEPROM MEMORY MAP**

| PIC18(L)F25K83    |                                                                   | PIC18(L)F26K83                              |          |
|-------------------|-------------------------------------------------------------------|---------------------------------------------|----------|
| PC<21:0>          |                                                                   | PC<21:0>                                    |          |
| ↕                 |                                                                   | ↕                                           |          |
| Stack (31 levels) |                                                                   | Stack (31 levels)                           |          |
| ↓                 |                                                                   | ↓                                           |          |
| 00 0000h          | Reset Vector                                                      | Reset Vector                                | 00 0000h |
| ...               | ...                                                               | ...                                         | ...      |
| 00 0008h          | Interrupt Vector High <sup>(2)</sup>                              | Interrupt Vector High <sup>(2)</sup>        | 00 0008h |
| ...               | ...                                                               | ...                                         | ...      |
| 00 0018h          | Interrupt Vector Low <sup>(2)</sup>                               | Interrupt Vector Low <sup>(2)</sup>         | 00 0018h |
| 00 001Ah          | Program Flash Memory (16 KW) <sup>(3)</sup>                       | Program Flash Memory (32 KW) <sup>(3)</sup> | 00 001Ah |
| 00 7FFFh          |                                                                   |                                             | 00 7FFFh |
| 00 8000h          |                                                                   |                                             | 00 8000h |
| 00 FFFFh          |                                                                   |                                             | 00 FFFFh |
| 01 0000h          | Not present <sup>(4)</sup>                                        | Not present <sup>(4)</sup>                  | 01 0000h |
| 1F FFFFh          |                                                                   |                                             | 1F FFFFh |
| 20 0000h          | User IDs (8 Words) <sup>(5)</sup>                                 |                                             | 20 0000h |
| ...               |                                                                   |                                             | ...      |
| 20 000Fh          |                                                                   |                                             | 20 000Fh |
| 20 0010h          | Reserved                                                          |                                             | 20 0010h |
| ...               |                                                                   |                                             | ...      |
| 2F FFFFh          |                                                                   |                                             | 2F FFFFh |
| 30 0000h          | Configuration Words (5 Words) <sup>(5)</sup>                      |                                             | 30 0000h |
| ...               |                                                                   |                                             | ...      |
| 30 0009h          |                                                                   |                                             | 30 0009h |
| 30 000Ah          | Reserved                                                          |                                             | 30 000Ah |
| ...               |                                                                   |                                             | ...      |
| 30 FFFFh          |                                                                   |                                             | 30 FFFFh |
| 31 0000h          | Data EEPROM (1024 Bytes)                                          |                                             | 31 0000h |
| ...               |                                                                   |                                             | ...      |
| 31 00FFh          |                                                                   |                                             | 31 00FFh |
| ...               |                                                                   |                                             | ...      |
| 31 0100h          |                                                                   |                                             | 31 0100h |
| ...               |                                                                   |                                             | ...      |
| 31 03FFh          |                                                                   |                                             | 31 03FFh |
| ...               |                                                                   |                                             | ...      |
| 31 0400h          | Reserved                                                          |                                             | 31 0400h |
| ...               |                                                                   |                                             | ...      |
| 3E FFFFh          |                                                                   |                                             | 3E FFFFh |
| 3F 0000h          | Device Information Area <sup>(5),(7)</sup>                        |                                             | 3F 0000h |
| ...               |                                                                   |                                             | ...      |
| 3F 003Fh          |                                                                   |                                             | 3F 003Fh |
| 3F0040h           | Reserved                                                          |                                             | 3F0040h  |
| ...               |                                                                   |                                             | ...      |
| 3F FEFFh          |                                                                   |                                             | 3F FEFFh |
| ...               |                                                                   |                                             | ...      |
| 3F FF00h          | Device Configuration Information (5 Words) <sup>(5),(6),(7)</sup> |                                             | 3F FF00h |
| ...               |                                                                   |                                             | ...      |
| 3F FF09h          |                                                                   |                                             | 3F FF09h |
| 3F FF0Ah          | Reserved                                                          |                                             | 3F FF0Ah |
| ...               |                                                                   |                                             | ...      |
| 3F FFFBh          |                                                                   |                                             | 3F FFFBh |
| ...               |                                                                   |                                             | ...      |
| 3F FFFCh          | Revision ID (1 Word) <sup>(5),(6),(7)</sup>                       |                                             | 3F FFFCh |
| ...               |                                                                   |                                             | ...      |
| 3F FFFDh          |                                                                   |                                             | 3F FFFDh |
| ...               |                                                                   |                                             | ...      |
| 3F FFFEh          | Device ID (1 Word) <sup>(5),(6),(7)</sup>                         |                                             | 3F FFFEh |
| ...               |                                                                   |                                             | ...      |
| 3F FFFFh          |                                                                   |                                             | 3F FFFFh |

**Note** 1: The stack is a separate SRAM panel, apart from all user memory panels.  
2: 00 0008h location is used as the reset default for the IVTBASE register, the vector table can be relocated in the memory by programming the IVTBASE register.  
3: Storage Area Flash is implemented as the last 128 Words of User Flash, if present.  
4: The addresses do not roll over. The region is read as '0'.  
5: Not code-protected.  
6: Hard-coded in silicon.  
7: This region cannot be written by the user and it is not affected by a Bulk Erase.



**TABLE 4-2: PROGRAM FLASH MEMORY PARTITION**

| Region               | Address                                                                                                                | Partition <sup>(3)</sup>                                      |                                                               |                                                               |                                                               |
|----------------------|------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------|---------------------------------------------------------------|---------------------------------------------------------------|---------------------------------------------------------------|
|                      |                                                                                                                        | $\overline{\text{BBEN}} = 1$<br>$\overline{\text{SAFEN}} = 1$ | $\overline{\text{BBEN}} = 1$<br>$\overline{\text{SAFEN}} = 0$ | $\overline{\text{BBEN}} = 0$<br>$\overline{\text{SAFEN}} = 1$ | $\overline{\text{BBEN}} = 0$<br>$\overline{\text{SAFEN}} = 0$ |
| Program Flash Memory | 00 0000h<br>• • •<br>Last Boot Block Memory Address                                                                    | APPLICATION BLOCK                                             | APPLICATION BLOCK                                             | BOOT BLOCK                                                    | BOOT BLOCK                                                    |
|                      | Last Boot Block Memory Address <sup>(1)</sup> + 1<br>• • •<br>Last Program Memory Address <sup>(2)</sup> - 100h        |                                                               |                                                               | APPLICATION BLOCK                                             | APPLICATION BLOCK                                             |
|                      | Last Program Memory Address <sup>(2)</sup> - FEh <sup>(4)</sup><br>• • •<br>Last Program Memory Address <sup>(2)</sup> |                                                               | STORAGE AREA FLASH                                            |                                                               | STORAGE AREA FLASH                                            |
|                      |                                                                                                                        |                                                               |                                                               |                                                               |                                                               |

**Note 1:** Last Boot Block Memory Address is based on BBSIZE<2:0>, see [Table 5-1](#).

**2:** For Last Program Memory Address, see [Table 5-1](#).

**3:** Refer to [Register 5-7](#): Configuration Word 4L for  $\overline{\text{BBEN}}$  and  $\overline{\text{SAFEN}}$  definitions.

**4:** Storage Area Flash is implemented as the last 128 Words of User Flash, if present.

## 4.2.4 PROGRAM COUNTER

The Program Counter (PC) specifies the address of the instruction to fetch for execution. The PC is 21-bit wide and is contained in three separate 8-bit registers. The low byte, known as the PCL register, is both readable and writable. The high byte, or PCH register, contains the PC<15:8> bits; it is not directly readable or writable. Updates to the PCH register are performed through the PCLATH register. The upper byte is called PCU. This register contains the PC<20:16> bits; it is also not directly readable or writable. Updates to the PCU register are performed through the PCLATU register.

The contents of PCLATH and PCLATU are transferred to the Program Counter by any operation that writes PCL. Similarly, the upper two bytes of the Program Counter are transferred to PCLATH and PCLATU by any operation that reads PCL. This is useful for computed offsets to the PC (see [Section 4.3.2.1 “Computed GOTO”](#)).

The PC addresses bytes in the program memory. To prevent the PC from becoming misaligned with word instructions, the Least Significant bit of PCL is fixed to a value of '0'. The PC increments by two to address sequential instructions in the program memory.

The `CALL`, `RCALL`, `GOTO` and program branch instructions write to the Program Counter directly. For these instructions, the contents of PCLATH and PCLATU are not transferred to the Program Counter.

## 4.2.5 RETURN ADDRESS STACK

The return address stack allows any combination of up to 31 program calls and interrupts to occur. The PC is pushed onto the stack when a `CALL` or `RCALL` instruction is executed or an interrupt is acknowledged. The PC value is pulled off the stack on a `RETURN`, `RETLW` or a `RETFIE` instruction. PCLATU and PCLATH are not affected by any of the `RETURN` or `CALL` instructions.

The stack operates as a 31-word by 21-bit RAM and a 5-bit Stack Pointer. The stack space is not part of either program or data space. The Stack Pointer is readable and writable and the address on the top of the stack is readable and writable through the Top-of-Stack (TOS) Special File Registers. Data can also be pushed to, or popped from the stack, using these registers.

A `CALL`, `CALLW` or `RCALL` instruction causes a push onto the stack; the Stack Pointer is first incremented and the location pointed to by the Stack Pointer is written with the contents of the PC (already pointing to the instruction following the `CALL`). A `RETURN` type instruction causes a pop from the stack; the contents of the location pointed to by the STKPTR are transferred to the PC and then the Stack Pointer is decremented.

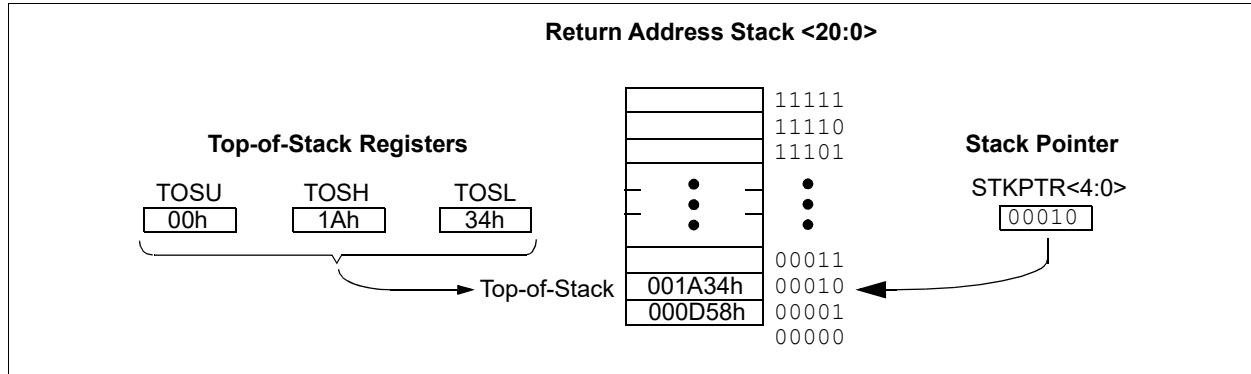
The Stack Pointer is initialized to '00000' after all Resets. There is no RAM associated with the location corresponding to a Stack Pointer value of '00000'; this is only a Reset value. Status bits in the PCON0 register indicate if the stack has overflowed or underflowed.

### 4.2.5.1 Top-of-Stack Access

Only the top of the return address stack (TOS) is readable and writable. A set of three registers, TOSU:TOSH:TOSL, holds the contents of the stack location pointed to by the STKPTR register ([Figure 4-1](#)). This allows users to implement a software stack, if necessary. After a `CALL`, `RCALL` or interrupt, the software can read the pushed value by reading the TOSU:TOSH:TOSL registers. These values can be placed on a user-defined software stack. At return time, the software can return these values to TOSU:TOSH:TOSL and do a return.

The user must disable the Global Interrupt Enable (GIE) bits while accessing the stack to prevent inadvertent stack corruption.

**FIGURE 4-1: RETURN ADDRESS STACK AND ASSOCIATED REGISTERS**



#### 4.2.5.2 Return Stack Pointer (STKPTR)

The STKPTR register (Register 4-4) contains the Stack Pointer value. The STKOVF (Stack Overflow) Status bit and the STKUNF (Stack Underflow) Status bit can be accessed using the PCON0 register. The value of the Stack Pointer can be 0 through 31. On Reset, the Stack Pointer value will be zero. The user may read and write the Stack Pointer value. This feature can be used by a Real-Time Operating System (RTOS) for stack maintenance. After the PC is pushed onto the stack 32 times (without popping any values off the stack), the STKOVF bit is set. The STKOVF bit is cleared by software or by a POR. The action that takes place when the stack becomes full depends on the state of the STVREN (Stack Overflow Reset Enable) Configuration bit. (Refer to Section 5.1 “Configuration Words” for a description of the device Configuration bits.)

If STVREN is set (default), a Reset will be generated and a Stack Overflow will be indicated by the STKOVF bit when the 32nd push is initiated. This includes CALL and CALLW instructions, as well as stacking the return address during an interrupt response. The STKOVF bit will remain set and the Stack Pointer will be set to zero.

If STVREN is cleared, the STKOVF bit will be set on the 32nd push and the Stack Pointer will remain at 31 but no Reset will occur. Any additional pushes will overwrite the 31<sup>st</sup> push but the STKPTR will remain at 31.

Setting STKOVF = 1 in software will change the bit, but will not generate a Reset.

The STKUNF bit is set when a stack pop returns a value of zero. The STKUNF bit is cleared by software or by POR. The action that takes place when the stack becomes full depends on the state of the STVREN (Stack Overflow Reset Enable) Configuration bit. (Refer to Section 5.1 “Configuration Words” for a description of the device Configuration bits.)

If STVREN is set (default) and the stack has been popped enough times to unload the stack, the next pop will return a value of zero to the PC, it will set the STKUNF bit and a Reset will be generated. This condition can be generated by the RETURN, RETLW and RETFIE instructions.

When STVREN = 0, STKUNF will be set but no Reset will occur.

**Note:** Returning a value of zero to the PC on an underflow has the effect of vectoring the program to the Reset vector, where the stack conditions can be verified and appropriate actions can be taken. This is not the same as a Reset, as the contents of the SFRs are not affected.

#### 4.2.5.3 PUSH and POP Instructions

Since the Top-of-Stack is readable and writable, the ability to push values onto the stack and pull values off the stack without disturbing normal program execution is a desirable feature. The PIC18 instruction set includes two instructions, PUSH and POP, that permit the TOS to be manipulated under software control. TOSU, TOSH and TOSL can be modified to place data or a return address on the stack.

The PUSH instruction places the current PC value onto the stack. This increments the Stack Pointer and loads the current PC value onto the stack.

The POP instruction discards the current TOS by decrementing the Stack Pointer. The previous value pushed onto the stack then becomes the TOS value.

## 4.3 Register Definitions: Stack Pointer

### REGISTER 4-1: TOSU: TOP-OF-STACK UPPER BYTE

|       |     |     |            |       |       |       |       |
|-------|-----|-----|------------|-------|-------|-------|-------|
| U-0   | U-0 | U-0 | R/W-0      | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| —     | —   | —   | TOS<20:16> |       |       |       |       |
| bit 7 |     |     |            |       |       |       | bit 0 |

#### Legend:

R = Readable bit      W = Writable bit      U = Unimplemented      C = Clearable only bit  
 -n = Value at POR      '1' = Bit is set      '0' = Bit is cleared      x = Bit is unknown

bit 7-5      **Unimplemented:** Read as '0'  
 bit 4-0      **TOS<20:16>:** Top-of-Stack Location bits

### REGISTER 4-2: TOSH: TOP-OF-STACK HIGH BYTE

|           |       |       |       |       |       |       |       |
|-----------|-------|-------|-------|-------|-------|-------|-------|
| R/W-0     | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| TOS<15:8> |       |       |       |       |       |       |       |
| bit 7     |       |       |       |       |       |       | bit 0 |

#### Legend:

R = Readable bit      W = Writable bit      U = Unimplemented      C = Clearable only bit  
 -n = Value at POR      '1' = Bit is set      '0' = Bit is cleared      x = Bit is unknown

bit 7-0      **TOS<15:8>:** Top-of-Stack Location bits

### REGISTER 4-3: TOSL: TOP-OF-STACK LOW BYTE

|          |       |       |       |       |       |       |       |
|----------|-------|-------|-------|-------|-------|-------|-------|
| R/W-0    | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| TOS<7:0> |       |       |       |       |       |       |       |
| bit 7    |       |       |       |       |       |       | bit 0 |

#### Legend:

R = Readable bit      W = Writable bit      U = Unimplemented      C = Clearable only bit  
 -n = Value at POR      '1' = Bit is set      '0' = Bit is cleared      x = Bit is unknown

bit 7-0      **TOS<7:0>:** Top-of-Stack Location bits

## REGISTER 4-4: STKPTR: STACK POINTER REGISTER

|       |     |     |             |       |       |       |       |
|-------|-----|-----|-------------|-------|-------|-------|-------|
| U-0   | U-0 | U-0 | R/W-0       | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| —     | —   | —   | STKPTR<4:0> |       |       |       |       |
| bit 7 |     |     |             |       |       |       | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented

C = Clearable only bit

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-5 **Unimplemented:** Read as '0'

bit 4-0 **STKPTR<4:0>:** Stack Pointer Location bits

### 4.3.1 FAST REGISTER STACK

There are three levels of fast stack registers available - one for `CALL` type instructions and two for interrupts. A fast register stack is provided for the Status, WREG and BSR registers, to provide a "fast return" option for interrupts. It is loaded with the current value of the corresponding register when the processor vectors for an interrupt. All interrupt sources will push values into the stack registers. The values in the registers are then loaded back into their associated registers if the `RETFIE`, `FAST` instruction is used to return from the interrupt. Refer to [Section 4.5.6 "Call Shadow Register"](#) for interrupt call shadow registers.

[Example 4-1](#) shows a source code example that uses the fast register stack during a subroutine call and return.

### EXAMPLE 4-1: FAST REGISTER STACK CODE EXAMPLE

```
CALL SUB1, FAST    ;STATUS, WREG, BSR
                   ;SAVED IN FAST REGISTER
                   ;STACK
    .
    .
SUB1    .
    .
    RETURN, FAST   ;RESTORE VALUES SAVED
                   ;IN FAST REGISTER STACK
```

## 4.3.2 LOOK-UP TABLES IN PROGRAM MEMORY

There may be programming situations that require the creation of data structures, or look-up tables, in program memory. For PIC18 devices, look-up tables can be implemented in two ways:

- Computed GOTO
- Table Reads

### 4.3.2.1 Computed GOTO

A computed GOTO is accomplished by adding an offset to the Program Counter. An example is shown in [Example 4-2](#).

A look-up table can be formed with an ADDWF PCL instruction and a group of RETLW nn instructions. The W register is loaded with an offset into the table before executing a call to that table. The first instruction of the called routine is the ADDWF PCL instruction. The next instruction executed will be one of the RETLW nn instructions that returns the value 'nn' to the calling function.

The offset value (in WREG) specifies the number of bytes that the Program Counter should advance and should be multiples of two (LSb = 0).

In this method, only one data byte may be stored in each instruction location and room on the return address stack is required.

#### EXAMPLE 4-2: COMPUTED GOTO USING AN OFFSET VALUE

|       |       |           |
|-------|-------|-----------|
|       | MOVWF | OFFSET, W |
|       | CALL  | TABLE     |
| ORG   | nn00h |           |
| TABLE | ADDWF | PCL       |
|       | RETLW | nnh       |
|       | RETLW | nnh       |
|       | RETLW | nnh       |
|       | .     |           |
|       | .     |           |
|       | .     |           |

### 4.3.2.2 Table Reads and Table Writes

A better method of storing data in program memory allows two bytes of data to be stored in each instruction location.

Look-up table data may be stored two bytes per program word by using table reads and writes. The Table Pointer (TBLPTR) register specifies the byte address and the Table Latch (TABLAT) register contains the data that is read from or written to program memory.

Table read and table write operations are discussed further in [Section 13.1.1 "Table Reads and Table Writes"](#).

## 4.4 PIC18 Instruction Cycle

### 4.4.1 CLOCKING SCHEME

The microcontroller clock input, whether from an internal or external source, is internally divided by four to generate four quadrature clocks (Q1, Q2, Q3 and Q4). Internally, the Program Counter is incremented on every Q1; the instruction is fetched from the program memory and latched into the instruction register during Q4. The instruction is decoded and executed during the following Q1 through Q4. The clocks and instruction execution flow are shown in [Figure 4-2](#).

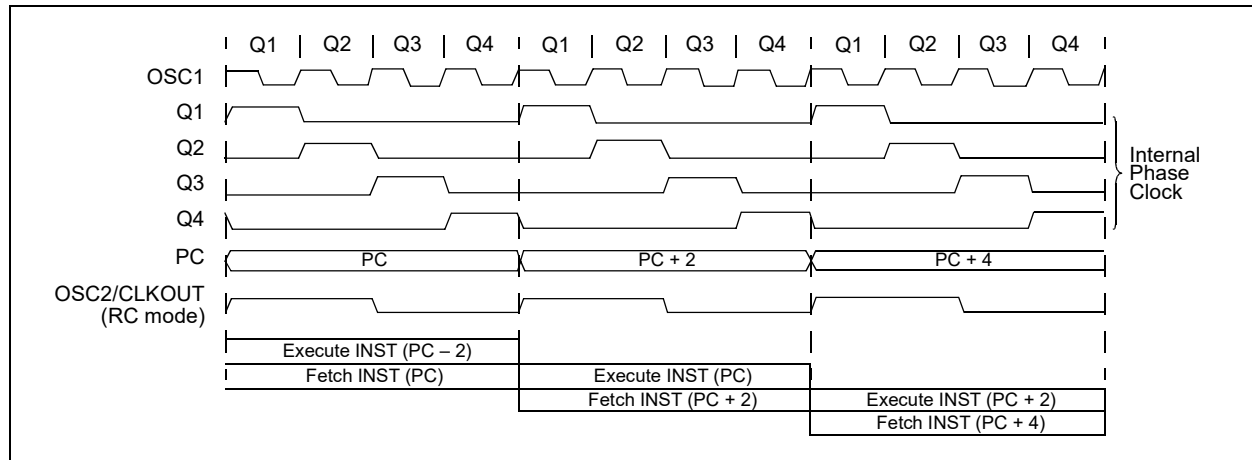
### 4.4.2 INSTRUCTION FLOW/PIPELINING

An “Instruction Cycle” consists of four Q cycles: Q1 through Q4. The instruction fetch and execute are pipelined in such a manner that a fetch takes one instruction cycle, while the decode and execute take another instruction cycle. However, due to the pipelining, each instruction effectively executes in one cycle. If an instruction causes the Program Counter to change (e.g., `GOTO`), then two cycles are required to complete the instruction ([Example 4-3](#)).

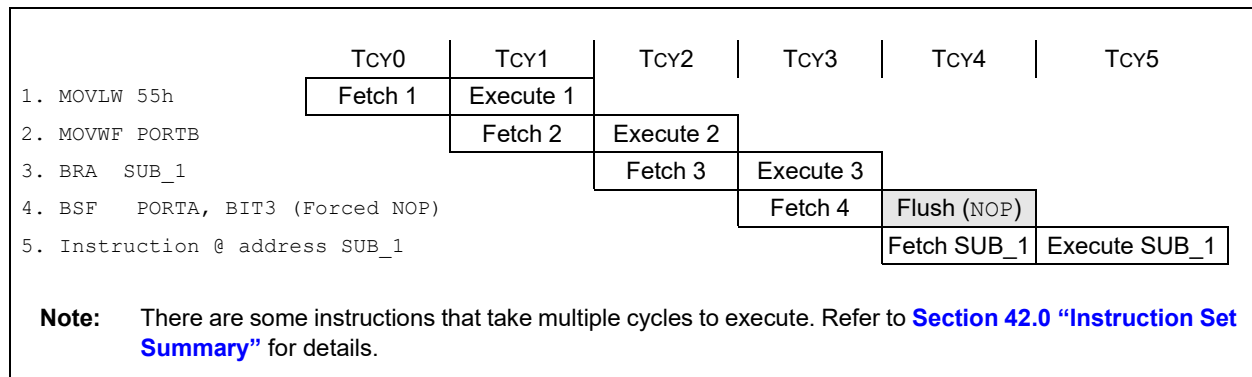
A fetch cycle begins with the Program Counter (PC) incrementing in Q1.

In the execution cycle, the fetched instruction is latched into the Instruction Register (IR) in cycle Q1. This instruction is then decoded and executed during the Q2, Q3 and Q4 cycles. Data memory is read during Q2 (operand read) and written during Q4 (destination write).

**FIGURE 4-2: CLOCK/INSTRUCTION CYCLE**



**EXAMPLE 4-3: INSTRUCTION PIPELINE FLOW**



## 4.4.3 INSTRUCTIONS IN PROGRAM MEMORY

The program memory is addressed in bytes. Instructions are stored as either two bytes or four bytes in program memory. The Least Significant Byte of an instruction word is always stored in a program memory location with an even address (LSb = 0). To maintain alignment with instruction boundaries, the PC increments in steps of two and the LSb will always read '0' (see [Section 4.2.4 “Program Counter”](#)).

[Figure 4-3](#) shows an example of how instruction words are stored in the program memory.

The `CALL` and `GOTO` instructions have the absolute program memory address embedded into the instruction. Since instructions are always stored on word boundaries, the data contained in the instruction is a word address. The word address is written to PC<20:1>, which accesses the desired byte address in program memory. Instruction #2 in [Figure 4-3](#) shows how the instruction `GOTO 0006h` is encoded in the program memory. Program branch instructions, which encode a relative address offset, operate in the same manner. The offset value stored in a branch instruction represents the number of single-word instructions that the PC will be offset by. [Section 42.0 “Instruction Set Summary”](#) provides further details of the instruction set.

## 4.4.4 MULTI-WORD INSTRUCTIONS

The standard PIC18 instruction set has four two-word instructions: `CALL`, `MOVFF`, `GOTO` and `LFSR` and two three-word instructions: `MOVFFL` and `MOVSL`. In all cases, the second and the third word of the instruction always has '1111' as its four Most Significant bits; the other 12 bits are literal data, usually a data memory address.

The use of '1111' in the four MSBs of an instruction specifies a special form of `NOP`. If the instruction is executed in proper sequence – immediately after the first word – the data in the second word is accessed and used by the instruction sequence. If the first word is skipped for some reason and the second or third word is executed by itself, a `NOP` is executed instead. This is necessary for cases when the multi-word instruction is preceded by a conditional instruction that changes the PC. [Example 4-4](#) shows how this works.

**FIGURE 4-3: INSTRUCTIONS IN PROGRAM MEMORY**

| Program Memory<br>Byte Locations → |        |            | LSB = 1 | LSB = 0 | Word Address<br>↓ |
|------------------------------------|--------|------------|---------|---------|-------------------|
|                                    |        |            |         |         |                   |
| Instruction 1:                     | MOVLW  | 055h       |         |         | 000000h           |
|                                    |        |            |         |         | 000002h           |
| Instruction 2:                     | GOTO   | 0006h      |         |         | 000004h           |
|                                    |        |            |         |         | 000006h           |
| Instruction 3:                     | MOVFF  | 123h, 456h | 0Fh     | 55h     | 000008h           |
|                                    |        |            | EFh     | 03h     | 00000Ah           |
|                                    |        |            | F0h     | 00h     | 00000Ch           |
|                                    |        |            | C1h     | 23h     | 00000Eh           |
| Instruction 4:                     | MOVFFL | 123h, 456h | F4h     | 56h     | 000010h           |
|                                    |        |            | 00h     | 60h     | 000012h           |
|                                    |        |            | F4h     | 8Ch     | 000014h           |
|                                    |        |            | F4h     | 56h     | 000016h           |
|                                    |        |            |         |         | 000018h           |
|                                    |        |            |         |         | 00001Ah           |



## EXAMPLE 4-4: TWO-WORD INSTRUCTIONS

| CASE 1:             |                                          |
|---------------------|------------------------------------------|
| Object Code         | Source Code                              |
| 0110 0110 0000 0000 | TSTFSZ REG1 ; is RAM location 0?         |
| 1100 0001 0010 0011 | MOVFF REG1, REG2 ; Yes, skip this word   |
| 1111 0100 0101 0110 | ; Execute this word as a NOP             |
| 0010 0100 0000 0000 | ADDWF REG3 ; continue code               |
| CASE 2:             |                                          |
| Object Code         | Source Code                              |
| 0110 0110 0000 0000 | TSTFSZ REG1 ; is RAM location 0?         |
| 1100 0001 0010 0011 | MOVFF REG1, REG2 ; No, execute this word |
| 1111 0100 0101 0110 | ; 2nd word of instruction                |
| 0010 0100 0000 0000 | ADDWF REG3 ; continue code               |

## EXAMPLE 4-5: THREE-WORD INSTRUCTIONS

| CASE 1:             |                                           |
|---------------------|-------------------------------------------|
| Object Code         | Source Code                               |
| 0110 0110 0000 0000 | TSTFSZ REG1 ; is RAM location 0?          |
| 0000 0000 0110 0000 | MOVFFL REG1, REG2 ; Yes, skip this word   |
| 1111 0100 1000 1100 | ; Execute this word as a NOP              |
| 1111 0100 0101 0110 | ; Execute this word as a NOP              |
| 0010 0100 0000 0000 | ADDWF REG3 ; continue code                |
| CASE 2:             |                                           |
| Object Code         | Source Code                               |
| 0110 0110 0000 0000 | TSTFSZ REG1 ; is RAM location 0?          |
| 0000 0000 0110 0000 | MOVFFL REG1, REG2 ; No, execute this word |
| 1111 0100 1000 1100 | ; 2nd word of instruction                 |
| 1111 0100 0101 0110 | ; 3rd word of instruction                 |
| 0010 0100 0000 0000 | ADDWF REG3 ; continue code                |

## 4.5 Data Memory Organization

Data memory in PIC18(L)F25/26K83 devices is implemented as static RAM. Each register in the data memory has a 14-bit address, allowing up to 16384 bytes of data memory. The memory space is divided into 64 banks that contain 256 bytes each. [Figure 4-5](#) shows the data memory organization for the PIC18(L)F25/26K83 devices in this data sheet.

The data memory contains Special Function Registers (SFRs) and General Purpose Registers (GPRs). The SFRs are used for control and status of the controller and peripheral functions, while GPRs are used for data storage and scratchpad operations in the user's application. Any read of an unimplemented location will read as '0's.

The instruction set and architecture allow operations across all banks. The entire data memory may be accessed by Direct, Indirect or Indexed Addressing modes. Addressing modes are discussed later in this subsection.

To ensure that commonly used registers (select SFRs and GPRs) can be accessed in a single cycle, PIC18 devices implement an Access Bank. This is a 256-byte memory space that provides fast access to some SFRs and the lower portion of GPR Bank 0 without using the Bank Select Register (BSR). [Section 4.5.4 "Access Bank"](#) provides a detailed description of the Access RAM.

### 4.5.1 BANK SELECT REGISTER (BSR)

Large areas of data memory require an efficient addressing scheme to make rapid access to any address possible. Ideally, this means that an entire address does not need to be provided for each read or write operation. For PIC18 devices, this is accomplished with a RAM banking scheme. This divides the memory space into 64 contiguous banks of 256 bytes. Depending on the instruction, each location can be addressed directly by its full 14-bit address, or an 8-bit low-order address and a 6-bit Bank Select Register.

This SFR holds the six Most Significant bits of a location address; the instruction itself includes the eight Least Significant bits. Only the six lower bits of the BSR are implemented (BSR<5:0>). The upper two bits are unused; they will always read '0' and cannot be written to. The BSR can be loaded directly by using the `MOVLB` instruction.

The value of the BSR indicates the bank in data memory; the eight bits in the instruction show the location in the bank and can be thought of as an offset from the bank's lower boundary. The relationship between the BSR's value and the bank division in data memory is shown in [Figure 4-5](#).

Since up to 64 registers may share the same low-order address, the user must always be careful to ensure that the proper bank is selected before performing a data read or write. For example, writing what should be program data to an 8-bit address of F9h while the BSR is 3Fh will end up corrupting the Program Counter.

While any bank can be selected, only those banks that are actually implemented can be read or written to. Writes to unimplemented banks are ignored, while reads from unimplemented banks will return '0's. Even so, the STATUS register will still be affected as if the operation was successful. The data memory maps in [Figure 4-5](#) indicate which banks are implemented.

# PIC18(L)F25/26K83

**FIGURE 4-4: DATA MEMORY MAP FOR PIC18(L)F25/26K83 DEVICES**

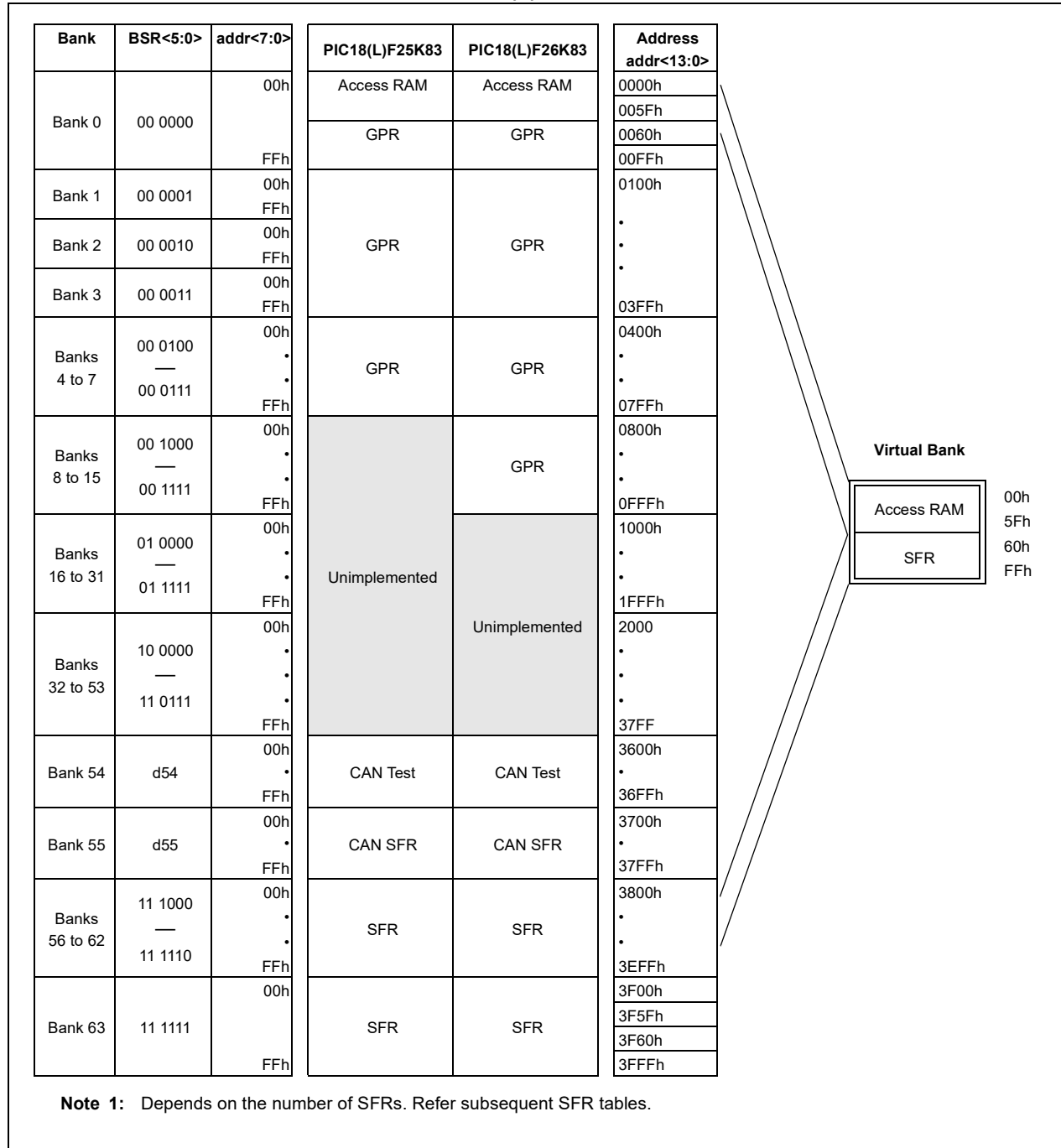
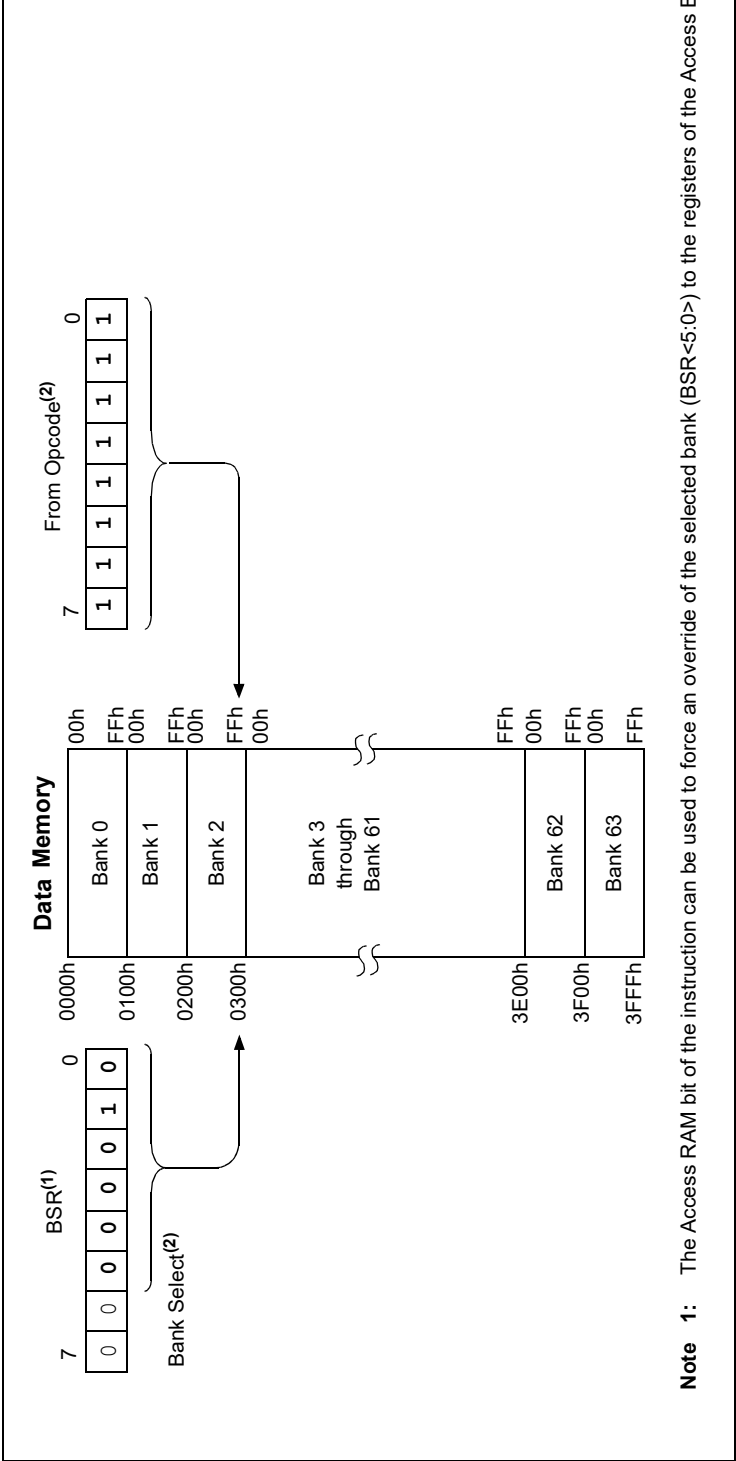


**FIGURE 4-5: USE OF THE BANK SELECT REGISTER (DIRECT ADDRESSING)**



## 4.5.2 GENERAL PURPOSE REGISTER FILE

General Purpose RAM is available starting Bank 0 of data memory. GPRs are not initialized by a Power-on Reset and are unchanged on all other Resets.

## 4.5.3 SPECIAL FUNCTION REGISTERS

The Special Function Registers (SFRs) are registers used by the CPU and peripheral modules for controlling the desired operation of the device. These registers are implemented as static RAM. SFRs start at the top of data memory (3FFFh) and extend downward to occupy Bank 56 through 63 (3800h to 3FFFh). A list of these registers is given in [Table 4-3](#) to [Table 4-10](#). A bitwise summary of these registers can be found in [Section 43.0 “Register Summary”](#).

## 4.5.4 ACCESS BANK

To streamline access for the most commonly used data memory locations, the data memory is configured with an Access Bank, which allows users to access a mapped block of memory without specifying a BSR. The Access Bank consists of the first 96 bytes of memory (00h-5Fh) in Bank 0 and the last 160 bytes of memory (60h-FFh) in Bank 63. The lower half is known as the “Access RAM” and is composed of GPRs. This upper half is also where some of the SFRs of the device are mapped. These two areas are mapped contiguously in the Access Bank and can be addressed linearly by an 8-bit address ([Figure 4-5](#)).

The Access Bank is used by core PIC18 instructions that include the Access RAM bit (the ‘a’ parameter in the instruction). When ‘a’ is equal to ‘1’, the instruction uses the BSR and the 8-bit address included in the opcode for the data memory address. When ‘a’ is ‘0’, however, the instruction uses the Access Bank address map; the current value of the BSR is ignored.

Using this “forced” addressing allows the instruction to operate on a data address in a single cycle, without updating the BSR first. For 8-bit addresses of 60h and above, this means that users can evaluate and operate on SFRs more efficiently. The Access RAM below 60h is a good place for data values that the user might need to access rapidly, such as immediate computational results or common program variables. Access RAM also allows for faster and more code efficient and switching of variables.

The mapping of the Access Bank is slightly different when the extended instruction set is enabled (XINST Configuration bit = 1). This is discussed in more detail in [Section 4.8.3 “Mapping the Access Bank in Indexed Literal Offset Mode”](#).

**TABLE 4-3: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F25/26K83 DEVICES BANK 63**

|        |          |       |          |      |        |       |        |       |         |       |          |       |       |
|--------|----------|-------|----------|------|--------|-------|--------|-------|---------|-------|----------|-------|-------|
| 3FFh   | TOSU     | 3DFh  | INDF2    | 3BFh | —      | 3F9Fh | T4PR   | 3F7Fh | GCP1CAP | 3F5Fh | CCPTMRS1 | 3F3Fh | NC01  |
| 3FEh   | TOSH     | 3DEh  | POSTINC2 | 3BEh | —      | 3F9Eh | T4TMR  | 3F7Eh | CCP1CON | 3F5Eh | CCPTMRS0 | 3F3Eh | NC01h |
| 3FDh   | TOSL     | 3DDh  | POSTDEC2 | 3BDh | —      | 3F9Dh | T5CLK  | 3F7Dh | CCPR1H  | 3F5Dh | —        | 3F3Dh | NC01h |
| 3FCCh  | STKPTR   | 3DCCh | PRECIN2  | 3BCh | LATC   | 3F9Ch | T5GATE | 3F7Ch | CCPR1L  | 3F5Ch | —        | 3F3Ch | NC01h |
| 3FBh   | PCLATU   | 3DBh  | PLUSW2   | 3BBh | LATB   | 3F9Bh | T5GCON | 3F7Bh | CCP2CAP | 3F5Bh | —        | 3F3Bh | NC01h |
| 3FAh   | PCLATH   | 3DBh  | FSR2H    | 3BAh | LATA   | 3F9Ah | T5CON  | 3F7Ah | CCP2CON | 3F5Ah | CWG1STR  | 3F3Ah | NC01A |
| 3FF9h  | PCL      | 3D9h  | FSR2L    | 3B9h | T0CON1 | 3F99h | TMR5H  | 3F79h | CCPR2H  | 3F59h | CWG1AS1  | 3F39h | NC01A |
| 3FF8h  | TBLPRTU  | 3D8h  | STATUS   | 3B8h | T0CON0 | 3F98h | TMR5L  | 3F78h | CCPR2L  | 3F58h | CWG1AS0  | 3F38h | NC01A |
| 3FF7h  | TBLPTRH  | 3D7h  | IVTBASEU | 3B7h | TMR0H  | 3F97h | T6RST  | 3F77h | CCP3CAP | 3F57h | CWG1CON1 | 3F37h | —     |
| 3FF6h  | TBLPTRL  | 3D6h  | IVTBASEH | 3B6h | TMR0L  | 3F96h | T6CLK  | 3F76h | CCP3CON | 3F56h | CWG1CON0 | 3F36h | —     |
| 3FF5h  | TABLAT   | 3D5h  | IVTBASEL | 3B5h | T1CLK  | 3F95h | T6HLT  | 3F75h | CCPR3H  | 3F55h | CWG1DBF  | 3F35h | —     |
| 3FF4h  | PRODH    | 3D4h  | IVTLOCK  | 3B4h | T1GATE | 3F94h | T6CON  | 3F74h | CCPR3L  | 3F54h | CWG1DBR  | 3F34h | —     |
| 3FF3h  | PRODL    | 3D3h  | INTCON1  | 3B3h | T1GCON | 3F93h | T6PR   | 3F73h | CCP4CAP | 3F53h | CWG1ISM  | 3F33h | —     |
| 3FF2h  | —        | 3D2h  | INTCON0  | 3B2h | T1CON  | 3F92h | T6TMR  | 3F72h | CCP4CON | 3F52h | CWG1CLK  | 3F32h | —     |
| 3FF1h  | PCON1    | 3D1h  | —        | 3B1h | TMR1H  | 3F91h | —      | 3F71h | CCPR4H  | 3F51h | CWG2STR  | 3F31h | —     |
| 3FF0h  | PCON0    | 3D0h  | —        | 3B0h | TMR1L  | 3F90h | —      | 3F70h | CCPR4L  | 3F50h | CWG2AS1  | 3F30h | —     |
| 3FEFh  | INDF0    | 3CFh  | —        | 3AFh | T2RST  | 3F8Fh | —      | 3F6Fh | —       | 3F4Fh | CWG2AS0  | 3F2Fh | —     |
| 3FEh   | POSTINC0 | 3CEh  | PORTE    | 3AEh | T2CLK  | 3F8Eh | —      | 3F6Eh | PWM5CON | 3F4Eh | CWG2CON1 | 3F2Eh | —     |
| 3FEDh  | POSTDEC0 | 3CDh  | —        | 3ADh | T2HLT  | 3F8Dh | —      | 3F6Dh | PWM5DCH | 3F4Dh | CWG2CON0 | 3F2Dh | —     |
| 3FECCh | PRECIN0  | 3CCCh | PORTC    | 3ACH | T2CON  | 3F8Ch | —      | 3F6Ch | PWM5DCL | 3F4Ch | CWG2DBF  | 3F2Ch | —     |
| 3FEBh  | PLUSW0   | 3CBh  | PORTB    | 3ABh | T2PR   | 3F8Bh | —      | 3F6Bh | —       | 3F4Bh | CWG2DBR  | 3F2Bh | —     |
| 3FEAh  | FSR0H    | 3CAh  | —        | 3AAh | T2TMR  | 3F8Ah | —      | 3F6Ah | PWM6CON | 3F4Ah | CWG2ISM  | 3F2Ah | —     |
| 3FE9h  | FSR0L    | 3C9h  | —        | 3A9h | T3CLK  | 3F89h | —      | 3F69h | PWM6DCH | 3F49h | CWG2CLK  | 3F29h | —     |
| 3FE8h  | WREG     | 3C8h  | —        | 3A8h | T3GATE | 3F88h | —      | 3F68h | PWM6DCL | 3F48h | CWG3STR  | 3F28h | —     |
| 3FE7h  | INDF1    | 3C7h  | —        | 3A7h | T3GCON | 3F87h | —      | 3F67h | —       | 3F47h | CWG3AS1  | 3F27h | —     |
| 3FE6h  | POSTINC1 | 3C6h  | —        | 3A6h | T3CON  | 3F86h | —      | 3F66h | PWM7CON | 3F46h | CWG3AS0  | 3F26h | —     |
| 3FE5h  | POSTDEC1 | 3C5h  | —        | 3A5h | TMR3H  | 3F85h | —      | 3F65h | PWM7DCH | 3F45h | CWG3CON1 | 3F25h | —     |
| 3FE4h  | PRECIN1  | 3C4h  | TRISC    | 3A4h | TMR3L  | 3F84h | —      | 3F64h | PWM7DCL | 3F44h | CWG3CON0 | 3F24h | —     |
| 3FE3h  | PLUSW1   | 3C3h  | TRISB    | 3A3h | T4RST  | 3F83h | —      | 3F63h | —       | 3F43h | CWG3DBF  | 3F23h | —     |
| 3FE2h  | FSR1H    | 3C2h  | TRISA    | 3A2h | T4CLK  | 3F82h | —      | 3F62h | PWM8CON | 3F42h | CWG3DBR  | 3F22h | —     |
| 3FE1h  | FSR1L    | 3C1h  | —        | 3A1h | T4HLT  | 3F81h | —      | 3F61h | PWM8DCH | 3F41h | CWG3ISM  | 3F21h | —     |
| 3FE0h  | BSR      | 3C0h  | —        | 3A0h | T4CON  | 3F80h | —      | 3F60h | PWM8DCL | 3F40h | CWG3CLK  | 3F20h | —     |

2: Unimplemented data memory locations and registers, read as '0'.

**TABLE 4-4: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F25/26K83 DEVICES BANK 62**

|       |         |       |          |       |         |       |          |       |   |       |   |       |   |
|-------|---------|-------|----------|-------|---------|-------|----------|-------|---|-------|---|-------|---|
| 3EFFh | ADCLK   | 3EDFh | ADLTHH   | 3EBFh | CM1PCH  | 3E9Fh | —        | 3E7Fh | — | 3E5Fh | — | 3E3Fh | — |
| 3EF0h | ADACT   | 3EDCh | ADLTHL   | 3EBCh | CM1INCH | 3E9Eh | DAC1CON0 | 3E7Eh | — | 3E5Eh | — | 3E3Eh | — |
| 3EF1h | ADREF   | 3EDBh | —        | 3EBBh | CM1CON1 | 3E9Dh | —        | 3E7Dh | — | 3E5Dh | — | 3E3Dh | — |
| 3EF2h | ADSTAT  | 3EDAh | —        | 3EBAh | CM2PCH  | 3E9Ch | DAC1CON1 | 3E7Ch | — | 3E5Ch | — | 3E3Ch | — |
| 3EF3h | ADCON3  | 3ED9h | —        | 3EBAh | CM2NCH  | 3E9Bh | —        | 3E7Bh | — | 3E5Bh | — | 3E3Bh | — |
| 3EF4h | ADCON2  | 3ED8h | —        | 3EB9h | CM2CON1 | 3E9Ah | —        | 3E7Ah | — | 3E5Ah | — | 3E3Ah | — |
| 3EF5h | ADCON1  | 3ED7h | —        | 3EB8h | CM2CON0 | 3E99h | —        | 3E79h | — | 3E59h | — | 3E39h | — |
| 3EF6h | ADCON0  | 3ED6h | —        | 3EB7h | —       | 3E98h | —        | 3E78h | — | 3E58h | — | 3E38h | — |
| 3EF7h | ADPREH  | 3ED5h | ADCP     | 3EB6h | —       | 3E97h | —        | 3E77h | — | 3E57h | — | 3E37h | — |
| 3EF8h | ADPREL  | 3ED4h | —        | 3EB5h | —       | 3E96h | —        | 3E76h | — | 3E56h | — | 3E36h | — |
| 3EF9h | ADCAP   | 3ED3h | —        | 3EB4h | —       | 3E95h | —        | 3E75h | — | 3E55h | — | 3E35h | — |
| 3EF0h | ADACQH  | 3ED2h | —        | 3EB3h | —       | 3E94h | —        | 3E74h | — | 3E54h | — | 3E34h | — |
| 3EF1h | ADACQL  | 3ED1h | —        | 3EB2h | —       | 3E93h | —        | 3E73h | — | 3E53h | — | 3E33h | — |
| 2EF2h | —       | 3ED0h | —        | 3EB1h | —       | 3E92h | —        | 3E72h | — | 3E52h | — | 3E32h | — |
| 3EF1h | ADPCH   | 3EDFh | —        | 3EAFh | —       | 3E91h | —        | 3E71h | — | 3E51h | — | 3E31h | — |
| 3EF0h | ADRESH  | 3EDCh | —        | 3EAEh | —       | 3E90h | —        | 3E70h | — | 3E50h | — | 3E30h | — |
| 3EF7h | ADRESL  | 3EDBh | —        | 3EADh | —       | 3E8Fh | —        | 3E6Fh | — | 3E4Fh | — | 3E2Fh | — |
| 3EEFh | ADPREVH | 3EDAh | —        | 3EABh | —       | 3E8Eh | —        | 3E6Eh | — | 3E4Eh | — | 3E2Eh | — |
| 3EE0h | ADPREVL | 3ED9h | —        | 3EACh | —       | 3E8Dh | —        | 3E6Dh | — | 3E4Dh | — | 3E2Dh | — |
| 3EE1h | ADRPT   | 3ED8h | —        | 3EABh | —       | 3E8Ch | —        | 3E6Ch | — | 3E4Ch | — | 3E2Ch | — |
| 3EE2h | ADCNT   | 3ED7h | —        | 3EAAh | —       | 3E8Bh | —        | 3E6Bh | — | 3E4Bh | — | 3E2Bh | — |
| 3EE3h | ADACCU  | 3ED6h | HLVDCON1 | 3EA9h | —       | 3E8Ah | —        | 3E6Ah | — | 3E4Ah | — | 3E2Ah | — |
| 3EE4h | ADACCH  | 3ED5h | HLVDCON0 | 3EA8h | —       | 3E89h | —        | 3E69h | — | 3E49h | — | 3E29h | — |
| 3EE5h | ADACCL  | 3ED4h | —        | 3EA7h | —       | 3E88h | —        | 3E68h | — | 3E48h | — | 3E28h | — |
| 3EE6h | ADFLTRH | 3ED3h | —        | 3EA6h | —       | 3E87h | —        | 3E67h | — | 3E47h | — | 3E27h | — |
| 3EE7h | ADFLTRL | 3ED2h | —        | 3EA5h | —       | 3E86h | —        | 3E66h | — | 3E46h | — | 3E26h | — |
| 3EE8h | ADSTPTH | 3ED1h | —        | 3EA4h | —       | 3E85h | —        | 3E65h | — | 3E45h | — | 3E25h | — |
| 3EE9h | ADSTPTL | 3ED0h | —        | 3EA3h | —       | 3E84h | —        | 3E64h | — | 3E44h | — | 3E24h | — |
| 3EE0h | ADERRH  | 3ECFh | ZCDCON   | 3EA2h | —       | 3E83h | —        | 3E63h | — | 3E43h | — | 3E23h | — |
| 3EE1h | ADERRL  | 3ECCh | —        | 3EA1h | —       | 3E82h | —        | 3E62h | — | 3E42h | — | 3E22h | — |
| 3EE2h | ADUTHH  | 3ECBh | FVRCON   | 3EAFh | —       | 3E81h | —        | 3E61h | — | 3E41h | — | 3E21h | — |
| 3EE3h | ADUTHL  | 3ECAh | CMOUT    | 3EA0h | —       | 3E80h | —        | 3E60h | — | 3E40h | — | 3E20h | — |

**Legend:** Unimplemented data memory locations and registers, read as '0'.





**TABLE 4-6: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F25/26K83 DEVICES BANK 60**

|       |         |       |       |   |       |   |       |   |       |          |       |
|-------|---------|-------|-------|---|-------|---|-------|---|-------|----------|-------|
| 3CFFh | —       | 3CDFh | 3CBFh | — | 3C9Fh | — | 3C7Fh | — | 3C5Fh | CLC4GLS3 | 3C3Fh |
| 3CFEh | MD1CARH | 3CDEh | 3CBEh | — | 3C9Eh | — | 3C7Eh | — | 3C5Eh | CLC4GLS2 | 3C3Eh |
| 3CFDh | MD1CARL | 3CDDh | 3CBDh | — | 3C9Dh | — | 3C7Dh | — | 3C5Dh | CLC4GLS1 | 3C3Dh |
| 3CFCh | MD1SRC  | 3CDC  | 3CBBh | — | 3C9Ch | — | 3C7Ch | — | 3C5Ch | CLC4GLS0 | 3C3Ch |
| 3CFBh | MD1CON1 | 3CDBh | 3CBBh | — | 3C9Bh | — | 3C7Bh | — | 3C5Bh | CLC4SEL3 | 3C3Bh |
| 3CFAh | MD1CON0 | 3CDAh | 3CBAh | — | 3C9Ah | — | 3C7Ah | — | 3C5Ah | CLC4SEL2 | 3C3Ah |
| 3CF9h | —       | 3CD9h | 3CB9h | — | 3C99h | — | 3C79h | — | 3C59h | CLC4SEL1 | 3C39h |
| 3CF8h | —       | 3CD8h | 3CB8h | — | 3C98h | — | 3C78h | — | 3C58h | CLC4SELO | 3C38h |
| 3CF7h | —       | 3CD7h | 3CB7h | — | 3C97h | — | 3C77h | — | 3C57h | CLC4POL  | 3C37h |
| 3CF6h | —       | 3CD6h | 3CB6h | — | 3C96h | — | 3C76h | — | 3C56h | CLC4CON  | 3C36h |
| 3CF5h | —       | 3CD5h | 3CB5h | — | 3C95h | — | 3C75h | — | 3C55h | —        | 3C35h |
| 3CF4h | —       | 3CD4h | 3CB4h | — | 3C94h | — | 3C74h | — | 3C54h | —        | 3C34h |
| 3CF3h | —       | 3CD3h | 3CB3h | — | 3C93h | — | 3C73h | — | 3C53h | —        | 3C33h |
| 3CF2h | —       | 3CD2h | 3CB2h | — | 3C92h | — | 3C72h | — | 3C52h | —        | 3C32h |
| 3CF1h | —       | 3CD1h | 3CB1h | — | 3C91h | — | 3C71h | — | 3C51h | —        | 3C31h |
| 3CF0h | —       | 3CD0h | 3CB0h | — | 3C90h | — | 3C70h | — | 3C50h | —        | 3C30h |
| 3CEFh | —       | 3CCFh | 3CAFh | — | 3C8Fh | — | 3C6Fh | — | 3C4Fh | —        | 3C2Fh |
| 3CEEh | —       | 3CCEh | 3CAEh | — | 3C8Eh | — | 3C6Eh | — | 3C4Eh | —        | 3C2Eh |
| 3CEDh | —       | 3CCDh | 3CADh | — | 3C8Dh | — | 3C6Dh | — | 3C4Dh | —        | 3C2Dh |
| 3CECh | —       | 3CCCh | 3CACH | — | 3C8Ch | — | 3C6Ch | — | 3C4Ch | —        | 3C2Ch |
| 3CEBh | —       | 3CCBh | 3CABh | — | 3C8Bh | — | 3C6Bh | — | 3C4Bh | —        | 3C2Bh |
| 3CEAh | —       | 3CCAh | 3CAAh | — | 3C8Ah | — | 3C6Ah | — | 3C4Ah | —        | 3C2Ah |
| 3CE9h | —       | 3CC9h | 3CA9h | — | 3C89h | — | 3C69h | — | 3C49h | —        | 3C29h |
| 3CE8h | —       | 3CC8h | 3CA8h | — | 3C88h | — | 3C68h | — | 3C48h | —        | 3C28h |
| 3CE7h | —       | 3CC7h | 3CA7h | — | 3C87h | — | 3C67h | — | 3C47h | —        | 3C27h |
| 3CE6h | CLKRCLK | 3CC6h | 3CA6h | — | 3C86h | — | 3C66h | — | 3C46h | —        | 3C26h |
| 3CE5h | CLKRCON | 3CC5h | 3CA5h | — | 3C85h | — | 3C65h | — | 3C45h | —        | 3C25h |
| 3CE4h | —       | 3CC4h | 3CA4h | — | 3C84h | — | 3C64h | — | 3C44h | —        | 3C24h |
| 3CE3h | —       | 3CC3h | 3CA3h | — | 3C83h | — | 3C63h | — | 3C43h | —        | 3C23h |
| 3CE2h | —       | 3CC2h | 3CA2h | — | 3C82h | — | 3C62h | — | 3C42h | —        | 3C22h |
| 3CE1h | —       | 3CC1h | 3CA1h | — | 3C81h | — | 3C61h | — | 3C41h | —        | 3C21h |
| 3CE0h | —       | 3CC0h | 3CA0h | — | 3C80h | — | 3C60h | — | 3C40h | —        | 3C20h |

**Legend:** Unimplemented data memory locations and registers, read as '0'.

**TABLE 4-7: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F25/26K83 DEVICES BANK 59**

|       |           |       |           |       |   |       |   |       |   |       |   |       |   |
|-------|-----------|-------|-----------|-------|---|-------|---|-------|---|-------|---|-------|---|
| 3BFh  | DMA1SIRQ  | 3BDFh | DMA2SIRQ  | 3BBFh | — | 3B9Fh | — | 3B7Fh | — | 3B5Fh | — | 3B3Fh | — |
| 3BEh  | DMA1AIRQ  | 3BDEh | DMA2AIRQ  | 3BBEh | — | 3B9Eh | — | 3B7Eh | — | 3B5Eh | — | 3B3Eh | — |
| 3BFDh | DMA1CON1  | 3BDDh | DMA2CON1  | 3BBDh | — | 3B9Dh | — | 3B7Dh | — | 3B5Dh | — | 3B3Dh | — |
| 3BFC  | DMA1CON0  | 3BDC  | DMA2CON0  | 3BBC  | — | 3B9C  | — | 3B7C  | — | 3B5C  | — | 3B3C  | — |
| 3BFBh | DMA1SSAU  | 3BDBh | DMA2SSAU  | 3BBBh | — | 3B9Bh | — | 3B7Bh | — | 3B5Bh | — | 3B3Bh | — |
| 3BFAh | DMA1SSAH  | 3BDAh | DMA2SSAH  | 3BBAh | — | 3B9Ah | — | 3B7Ah | — | 3B5Ah | — | 3B3Ah | — |
| 3BF9h | DMA1SSAL  | 3BD9h | DMA2SSAL  | 3BB9h | — | 3B99h | — | 3B79h | — | 3B59h | — | 3B39h | — |
| 3BF8h | DMA1SSZH  | 3BD8h | DMA2SSZH  | 3BB8h | — | 3B98h | — | 3B78h | — | 3B58h | — | 3B38h | — |
| 3BF7h | DMA1SSZL  | 3BD7h | DMA2SSZL  | 3BB7h | — | 3B97h | — | 3B77h | — | 3B57h | — | 3B37h | — |
| 3BF6h | DMA1SPTRU | 3BD6h | DMA2SPTRU | 3BB6h | — | 3B96h | — | 3B76h | — | 3B56h | — | 3B36h | — |
| 3BF5h | DMA1SPTRH | 3BD5h | DMA2SPTRH | 3BB5h | — | 3B95h | — | 3B75h | — | 3B55h | — | 3B35h | — |
| 3BF4h | DMA1SPTRL | 3BD4h | DMA2SPTRL | 3BB4h | — | 3B94h | — | 3B74h | — | 3B54h | — | 3B34h | — |
| 3BF3h | DMA1SCNTH | 3BD3h | DMA2SCNTH | 3BB3h | — | 3B93h | — | 3B73h | — | 3B53h | — | 3B33h | — |
| 3BF2h | DMA1SCNTL | 3BD2h | DMA2SCNTL | 3BB2h | — | 3B92h | — | 3B72h | — | 3B52h | — | 3B32h | — |
| 3BF1h | DMA1DSAH  | 3BD1h | DMA2DSAH  | 3BB1h | — | 3B91h | — | 3B71h | — | 3B51h | — | 3B31h | — |
| 3BF0h | DMA1DSAL  | 3BD0h | DMA2DSAL  | 3BB0h | — | 3B90h | — | 3B70h | — | 3B50h | — | 3B30h | — |
| 3BEFh | DMA1DSZH  | 3BCFh | DMA2DSZH  | 3BAFh | — | 3B8Fh | — | 3B6Fh | — | 3B4Fh | — | 3B2Fh | — |
| 3BEEh | DMA1DSZL  | 3BCEh | DMA2DSZL  | 3BAEh | — | 3B8Eh | — | 3B6Eh | — | 3B4Eh | — | 3B2Eh | — |
| 3BEDh | DMA1DPTRH | 3BCDh | DMA2DPTRH | 3BADh | — | 3B8Dh | — | 3B6Dh | — | 3B4Dh | — | 3B2Dh | — |
| 3BEC  | DMA1DPTRL | 3BCCh | DMA2DPTRL | 3BACH | — | 3B8Ch | — | 3B6Ch | — | 3B4Ch | — | 3B2Ch | — |
| 3BEBh | DMA1DCNTH | 3BCBh | DMA2DCNTH | 3BABh | — | 3B8Bh | — | 3B6Bh | — | 3B4Bh | — | 3B2Bh | — |
| 3BEAh | DMA1DCNTL | 3BCAh | DMA2DCNTL | 3BAAh | — | 3B8Ah | — | 3B6Ah | — | 3B4Ah | — | 3B2Ah | — |
| 3BE9h | DMA1BUF   | 3BC9h | DMA2BUF   | 3BA9h | — | 3B89h | — | 3B69h | — | 3B49h | — | 3B29h | — |
| 3BE8h | —         | 3BC8h | —         | 3BA8h | — | 3B88h | — | 3B68h | — | 3B48h | — | 3B28h | — |
| 3BE7h | —         | 3BC7h | —         | 3BA7h | — | 3B87h | — | 3B67h | — | 3B47h | — | 3B27h | — |
| 3BE6h | —         | 3BC6h | —         | 3BA6h | — | 3B86h | — | 3B66h | — | 3B46h | — | 3B26h | — |
| 3BE5h | —         | 3BC5h | —         | 3BA5h | — | 3B85h | — | 3B65h | — | 3B45h | — | 3B25h | — |
| 3BE4h | —         | 3BC4h | —         | 3BA4h | — | 3B84h | — | 3B64h | — | 3B44h | — | 3B24h | — |
| 3BE3h | —         | 3BC3h | —         | 3BA3h | — | 3B83h | — | 3B63h | — | 3B43h | — | 3B23h | — |
| 3BE2h | —         | 3BC2h | —         | 3BA2h | — | 3B82h | — | 3B62h | — | 3B42h | — | 3B22h | — |
| 3BE1h | —         | 3BC1h | —         | 3BA1h | — | 3B81h | — | 3B61h | — | 3B41h | — | 3B21h | — |
| 3BE0h | —         | 3BC0h | —         | 3BA0h | — | 3B80h | — | 3B60h | — | 3B40h | — | 3B20h | — |

**Legend:** Unimplemented data memory locations and registers, read as '0'.

**TABLE 4-8: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F25/26K83 DEVICES BANK 58**

|        |            |        |            |       |         |       |   |       |   |       |   |       |   |
|--------|------------|--------|------------|-------|---------|-------|---|-------|---|-------|---|-------|---|
| 3AFFh  | —          | 3ADFh  | ADACTPPS   | 3ABFh | PPSLOCK | 3A9Fh | — | 3A7Fh | — | 3A5Fh | — | 3A3Fh | — |
| 3AFEh  | —          | 3ADCh  | CLCIN3PPS  | 3ABCh | —       | 3A9Eh | — | 3A7Eh | — | 3A5Eh | — | 3A3Eh | — |
| 3AFDh  | —          | 3ADDh  | CLCIN2PPS  | 3ABDh | —       | 3A9Dh | — | 3A7Dh | — | 3A5Dh | — | 3A3Dh | — |
| 3AFCCh | —          | 3ADCh  | CLCIN1PPS  | 3ABCh | —       | 3A9Ch | — | 3A7Ch | — | 3A5Ch | — | 3A3Ch | — |
| 3AFBh  | —          | 3ADBh  | CLCIN0PPS  | 3ABh  | —       | 3A9Bh | — | 3A7Bh | — | 3A5Bh | — | 3A3Bh | — |
| 3AFAh  | —          | 3ADAh  | MD1SRCPPS  | 3ABAh | —       | 3A9Ah | — | 3A7Ah | — | 3A5Ah | — | 3A3Ah | — |
| 3AF9h  | —          | 3AD9h  | MD1CARHPPS | 3AB9h | —       | 3A99h | — | 3A79h | — | 3A59h | — | 3A39h | — |
| 3AF8h  | —          | 3AD8h  | MD1CARLPPS | 3AB8h | —       | 3A98h | — | 3A78h | — | 3A58h | — | 3A38h | — |
| 3AF7h  | —          | 3AD7h  | CWG3INPPS  | 3AB7h | —       | 3A97h | — | 3A77h | — | 3A57h | — | 3A37h | — |
| 3AF6h  | —          | 3AD6h  | CWG2INPPS  | 3AB6h | —       | 3A96h | — | 3A76h | — | 3A56h | — | 3A36h | — |
| 3AF5h  | —          | 3AD5h  | CWG1INPPS  | 3AB5h | —       | 3A95h | — | 3A75h | — | 3A55h | — | 3A35h | — |
| 3AF4h  | —          | 3AD4h  | SMT2SIGPPS | 3AB4h | —       | 3A94h | — | 3A74h | — | 3A54h | — | 3A34h | — |
| 3AF3h  | —          | 3AD3h  | SMT2WINPPS | 3AB3h | —       | 3A93h | — | 3A73h | — | 3A53h | — | 3A33h | — |
| 3AF2h  | —          | 3AD2h  | SMT1SIGPPS | 3AB2h | —       | 3A92h | — | 3A72h | — | 3A52h | — | 3A32h | — |
| 3AF1h  | —          | 3AD1h  | SMT1WINPPS | 3AB1h | —       | 3A91h | — | 3A71h | — | 3A51h | — | 3A31h | — |
| 3AF0h  | —          | 3AD0h  | CCP4PPS    | 3AB0h | —       | 3A90h | — | 3A70h | — | 3A50h | — | 3A30h | — |
| 3AEFh  | —          | 3ACFh  | GCP3PPS    | 3AAFh | —       | 3A8Fh | — | 3A6Fh | — | 3A4Fh | — | 3A2Fh | — |
| 3AEEh  | —          | 3ACEh  | CCP2PPS    | 3AAEh | —       | 3A8Eh | — | 3A6Eh | — | 3A4Eh | — | 3A2Eh | — |
| 3AEDh  | CANRXPPS   | 3ACDh  | GCP1PPS    | 3AADh | —       | 3A8Dh | — | 3A6Dh | — | 3A4Dh | — | 3A2Dh | — |
| 3AECCh | —          | 3ACCCh | T6INPPS    | 3AACH | —       | 3A8Ch | — | 3A6Ch | — | 3A4Ch | — | 3A2Ch | — |
| 3AEBh  | U2CTSPPS   | 3ACBh  | T4INPPS    | 3AABh | —       | 3A8Bh | — | 3A6Bh | — | 3A4Bh | — | 3A2Bh | — |
| 3AEACh | U2RXPPS    | 3ACACh | T2INPPS    | 3AACh | —       | 3A8Ah | — | 3A6Ah | — | 3A4Ah | — | 3A2Ah | — |
| 3AE9h  | —          | 3AC9h  | T5GPPS     | 3AA9h | —       | 3A89h | — | 3A69h | — | 3A49h | — | 3A29h | — |
| 3AE8h  | U1CTSPPS   | 3AC8h  | T5CKIPPS   | 3AA8h | —       | 3A88h | — | 3A68h | — | 3A48h | — | 3A28h | — |
| 3AE7h  | U1RXPPS    | 3AC7h  | T3GPPS     | 3AA7h | —       | 3A87h | — | 3A67h | — | 3A47h | — | 3A27h | — |
| 3AE6h  | I2C2SDAPPS | 3AC6h  | T3CKIPPS   | 3AA6h | —       | 3A86h | — | 3A66h | — | 3A46h | — | 3A26h | — |
| 3AE5h  | I2C2SCLPPS | 3AC5h  | T1GPPS     | 3AA5h | —       | 3A85h | — | 3A65h | — | 3A45h | — | 3A25h | — |
| 3AE4h  | I2C1SDAPPS | 3AC4h  | T1CKIPPS   | 3AA4h | —       | 3A84h | — | 3A64h | — | 3A44h | — | 3A24h | — |
| 3AE3h  | I2C1SCLPPS | 3AC3h  | T0CKIPPS   | 3AA3h | —       | 3A83h | — | 3A63h | — | 3A43h | — | 3A23h | — |
| 3AE2h  | SPI1SSPPS  | 3AC2h  | INT2PPS    | 3AA2h | —       | 3A82h | — | 3A62h | — | 3A42h | — | 3A22h | — |
| 3AE1h  | SPI1SDIPPS | 3AC1h  | INT1PPS    | 3AA1h | —       | 3A81h | — | 3A61h | — | 3A41h | — | 3A21h | — |
| 3AE0h  | SPI1SCKPPS | 3AC0h  | INT0PPS    | 3AA0h | —       | 3A80h | — | 3A60h | — | 3A40h | — | 3A20h | — |

**Legend:** Unimplemented data memory locations and registers, read as '0'.

**TABLE 4-9: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F25/26K83 DEVICES BANK 57**

|       |         |       |            |       |      |       |      |       |           |       |         |       |
|-------|---------|-------|------------|-------|------|-------|------|-------|-----------|-------|---------|-------|
| 39FFh | —       | 39DFh | OSCFRQ     | 39BFh | —    | 399Fh | —    | 397Fh | —         | 395Fh | WDTU    | 393Fh |
| 39FEh | —       | 39DEh | OSCTUNE    | 39BEh | —    | 399Eh | —    | 397Eh | —         | 395Eh | WDTH    | 393Eh |
| 39FDh | —       | 39DDh | OSCEN      | 39BDh | —    | 399Dh | —    | 397Dh | SCANTRIG  | 395Dh | WDTL    | 393Dh |
| 39FCh | —       | 39DCh | OSCSTAT    | 39BCh | —    | 399Ch | —    | 397Ch | SCANCON0  | 395Ch | WDTCON1 | 393Ch |
| 39FBh | —       | 39DBh | OSCON3     | 39BBh | —    | 399Bh | —    | 397Bh | SCANHADRH | 395Bh | WDTCON0 | 393Bh |
| 39FAh | —       | 39DAh | OSCON2     | 39BAh | —    | 399Ah | —    | 397Ah | SCANHADRH | 395Ah | —       | 393Ah |
| 39F9h | —       | 39D9h | OSCON1     | 39B9h | —    | 3999h | PIE9 | 3979h | SCANHADRH | 3959h | —       | 3939h |
| 39F8h | —       | 39D8h | CPUDOZE    | 39B8h | —    | 3998h | PIE8 | 3978h | SCANLADRU | 3958h | —       | 3938h |
| 39F7h | SCANPR  | 39D7h | —          | 39B7h | —    | 3997h | PIE7 | 3977h | SCANLADRH | 3957h | —       | 3937h |
| 39F6h | —       | 39D6h | —          | 39B6h | —    | 3996h | PIE6 | 3976h | SCANLADRL | 3956h | —       | 3936h |
| 39F5h | —       | 39D5h | —          | 39B5h | —    | 3995h | PIE5 | 3975h | —         | 3955h | —       | 3935h |
| 39F4h | DMA2PR  | 39D4h | —          | 39B4h | —    | 3994h | PIE4 | 3974h | —         | 3954h | —       | 3934h |
| 39F3h | DMA1PR  | 39D3h | —          | 39B3h | —    | 3993h | PIE3 | 3973h | —         | 3953h | —       | 3933h |
| 39F2h | MAINPR  | 39D2h | —          | 39B2h | —    | 3992h | PIE2 | 3972h | —         | 3952h | —       | 3932h |
| 39F1h | ISRPR   | 39D1h | VREGCON(*) | 39B1h | —    | 3991h | PIE1 | 3971h | —         | 3951h | —       | 3931h |
| 39F0h | —       | 39D0h | BORCON     | 39B0h | —    | 3990h | PIE0 | 3970h | —         | 3950h | —       | 3930h |
| 39EFh | PRLOCK  | 39CFh | —          | 39AFh | —    | 398Fh | —    | 396Fh | —         | 394Fh | —       | 392Fh |
| 39EEh | —       | 39CEh | —          | 39AEh | —    | 398Eh | —    | 396Eh | —         | 394Eh | —       | 392Eh |
| 39EDh | —       | 39CDh | —          | 39ADh | —    | 398Dh | —    | 396Dh | —         | 394Dh | —       | 392Dh |
| 39ECh | —       | 39CCh | —          | 39ACh | —    | 398Ch | —    | 396Ch | —         | 394Ch | —       | 392Ch |
| 39EBh | —       | 39CBh | —          | 39ABh | —    | 398Bh | —    | 396Bh | —         | 394Bh | —       | 392Bh |
| 39EAh | —       | 39CAh | —          | 39AAh | —    | 398Ah | —    | 396Ah | —         | 394Ah | —       | 392Ah |
| 39E9h | —       | 39C9h | —          | 39A9h | PIR9 | 3989h | IPR9 | 3969h | CRCCON1   | 3949h | —       | 3929h |
| 39E8h | —       | 39C8h | —          | 39A8h | PIR8 | 3988h | IPR8 | 3968h | CRCCON0   | 3948h | —       | 3928h |
| 39E7h | —       | 39C7h | PMD7       | 39A7h | PIR7 | 3987h | IPR7 | 3967h | CRCXORH   | 3947h | —       | 3927h |
| 39E6h | NVMCON2 | 39C6h | PMD6       | 39A6h | PIR6 | 3986h | IPR6 | 3966h | CRCXORL   | 3946h | —       | 3926h |
| 39E5h | NVMCON1 | 39C5h | PMD5       | 39A5h | PIR5 | 3985h | IPR5 | 3965h | CRCSHIFTH | 3945h | —       | 3925h |
| 39E4h | —       | 39C4h | PMD4       | 39A4h | PIR4 | 3984h | IPR4 | 3964h | CRCSHIFTL | 3944h | —       | 3924h |
| 39E3h | NVMDAT  | 39C3h | PMD3       | 39A3h | PIR3 | 3983h | IPR3 | 3963h | CRCACCH   | 3943h | —       | 3923h |
| 39E2h | —       | 39C2h | PMD2       | 39A2h | PIR2 | 3982h | IPR2 | 3962h | CRCACCL   | 3942h | —       | 3922h |
| 39E1h | —       | 39C1h | PMD1       | 39A1h | PIR1 | 3981h | IPR1 | 3961h | CRCDATL   | 3941h | —       | 3921h |
| 39E0h | NVMADRL | 39C0h | PMD0       | 39A0h | PIR0 | 3980h | IPR0 | 3960h | —         | 3940h | —       | 3920h |

**Legend:** Unimplemented data memory locations and registers, read as '0'.

**Note 1:** Unimplemented in LF devices.

**TABLE 4-10: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F25/26K83 DEVICES BANK 56**

|       |   |       |   |       |   |       |              |       |       |   |       |   |
|-------|---|-------|---|-------|---|-------|--------------|-------|-------|---|-------|---|
| 38FFh | — | 38DFh | — | 38BFh | — | 389Fh | IVTADU       | 387Fh | 385Fh | — | 383Fh | — |
| 38FEh | — | 38DEh | — | 38BEh | — | 389Eh | IVTADH       | 387Eh | 385Eh | — | 383Eh | — |
| 38FDh | — | 38DDh | — | 38BDh | — | 389Dh | IVTADL       | 387Dh | 385Dh | — | 383Dh | — |
| 38FCh | — | 38DCh | — | 38BCh | — | 389Ch | —            | 387Ch | 385Ch | — | 383Ch | — |
| 38FBh | — | 38DBh | — | 38BBh | — | 389Bh | —            | 387Bh | 385Bh | — | 383Bh | — |
| 38FAh | — | 38DAh | — | 38BAh | — | 389Ah | —            | 387Ah | 385Ah | — | 383Ah | — |
| 38F9h | — | 38D9h | — | 38B9h | — | 3899h | —            | 3879h | 3859h | — | 3839h | — |
| 38F8h | — | 38D8h | — | 38B8h | — | 3898h | —            | 3878h | 3858h | — | 3838h | — |
| 38F7h | — | 38D7h | — | 38B7h | — | 3897h | —            | 3877h | 3857h | — | 3837h | — |
| 38F6h | — | 38D6h | — | 38B6h | — | 3896h | —            | 3876h | 3856h | — | 3836h | — |
| 38F5h | — | 38D5h | — | 38B5h | — | 3895h | —            | 3875h | 3855h | — | 3835h | — |
| 38F4h | — | 38D4h | — | 38B4h | — | 3894h | —            | 3874h | 3854h | — | 3834h | — |
| 38F3h | — | 38D3h | — | 38B3h | — | 3893h | —            | 3873h | 3853h | — | 3833h | — |
| 38F2h | — | 38D2h | — | 38B2h | — | 3892h | —            | 3872h | 3852h | — | 3832h | — |
| 38F1h | — | 38D1h | — | 38B1h | — | 3891h | —            | 3871h | 3851h | — | 3831h | — |
| 38F0h | — | 38D0h | — | 38B0h | — | 3890h | PRODH_SHAD   | 3870h | 3850h | — | 3830h | — |
| 38EFh | — | 38CFh | — | 38AFh | — | 388Fh | PRODL_SHAD   | 386Fh | 384Fh | — | 382Fh | — |
| 38EEh | — | 38CEh | — | 38AEh | — | 388Eh | FSR2H_SHAD   | 386Eh | 384Eh | — | 382Eh | — |
| 38EDh | — | 38CDh | — | 38ADh | — | 388Dh | FSR2L_SHAD   | 386Dh | 384Dh | — | 382Dh | — |
| 38ECh | — | 38CCh | — | 38ACH | — | 388Ch | FSR1H_SHAD   | 386Ch | 384Ch | — | 382Ch | — |
| 38EBh | — | 38CBh | — | 38ABh | — | 388Bh | FSR1L_SHAD   | 386Bh | 384Bh | — | 382Bh | — |
| 38EAh | — | 38CAh | — | 38AAh | — | 388Ah | FSR0H_SHAD   | 386Ah | 384Ah | — | 382Ah | — |
| 38E9h | — | 38C9h | — | 38A9h | — | 3889h | FSR0L_SHAD   | 3869h | 3849h | — | 3829h | — |
| 38E8h | — | 38C8h | — | 38A8h | — | 3888h | PCLATU_SHAD  | 3868h | 3848h | — | 3828h | — |
| 38E7h | — | 38C7h | — | 38A7h | — | 3887h | PCLATH_SHAD  | 3867h | 3847h | — | 3827h | — |
| 38E6h | — | 38C6h | — | 38A6h | — | 3886h | BSR_SHAD     | 3866h | 3846h | — | 3826h | — |
| 38E5h | — | 38C5h | — | 38A5h | — | 3885h | WREG_SHAD    | 3865h | 3845h | — | 3825h | — |
| 38E4h | — | 38C4h | — | 38A4h | — | 3884h | STATUS_SHAD  | 3864h | 3844h | — | 3824h | — |
| 38E3h | — | 38C3h | — | 38A3h | — | 3883h | SHADCON      | 3863h | 3843h | — | 3823h | — |
| 38E2h | — | 38C2h | — | 38A2h | — | 3882h | BSR_CSHAD    | 3862h | 3842h | — | 3822h | — |
| 38E1h | — | 38C1h | — | 38A1h | — | 3881h | WREG_CSHAD   | 3861h | 3841h | — | 3821h | — |
| 38E0h | — | 38C0h | — | 38A0h | — | 3880h | STATUS_CSHAD | 3860h | 3840h | — | 3820h | — |

**Legend:** Unimplemented data memory locations and registers, read as '0'.

**TABLE 4-11: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F25/26K83 DEVICES BANK 55**

|       |             |       |             |       |          |       |             |       |             |       |             |       |          |
|-------|-------------|-------|-------------|-------|----------|-------|-------------|-------|-------------|-------|-------------|-------|----------|
| 37FFh | CANCON_RO0  | 37DFh | CANCON_RO2  | 37BFh | RXM1EIDL | 379Fh | CANSTAT_RO4 | 377Fh | CANCON_RO6  | 375Fh | CANSTAT_RO8 | 373Fh | TXB1EIDL |
| 37FEh | CANSTAT_RO0 | 37DEh | CANSTAT_RO2 | 37BEh | RXM1EIDL | 379Eh | CANSTAT_RO4 | 377Eh | CANSTAT_RO6 | 375Eh | CANSTAT_RO8 | 373Eh | BIE      |
| 37FDh | RXB1D7      | 37DDh | TXB1D7      | 37BDh | RXM1SIDL | 379Dh | B5D7        | 377Dh | B3D7        | 375Dh | B1D7        | 373Dh | B5E      |
| 37FC  | RXB1D6      | 37DCh | TXB1D6      | 37BCh | RXM1SIDH | 379Ch | B5D6        | 377Ch | B3D6        | 375Ch | B1D6        | 373Ch | M5E      |
| 37FBh | RXB1D5      | 37DBh | TXB1D5      | 37BBh | RXM0EIDL | 379Bh | B5D5        | 377Bh | B3D5        | 375Bh | B1D5        | 373Bh | M5E      |
| 37FAh | RXB1D4      | 37DAh | TXB1D4      | 37BAh | RXM0EIDL | 379Ah | B5D4        | 377Ah | B3D4        | 375Ah | B1D4        | 373Ah | M5E      |
| 37F9h | RXB1D3      | 37D9h | TXB1D3      | 37B9h | RXM0SIDL | 3799h | B5D3        | 3779h | B3D3        | 3759h | B1D3        | 3739h | M5E      |
| 37F8h | RXB1D2      | 37D8h | TXB1D2      | 37B8h | RXM0SIDH | 3798h | B5D2        | 3778h | B3D2        | 3758h | B1D2        | 3738h | RXFBC    |
| 37F7h | RXB1D1      | 37D7h | TXB1D1      | 37B7h | RXF5EIDL | 3797h | B5D1        | 3777h | B3D1        | 3757h | B1D1        | 3737h | RXFBC    |
| 37F6h | RXB1D0      | 37D6h | TXB1D0      | 37B6h | RXF5EIDL | 3796h | B5D0        | 3776h | B3D0        | 3756h | B1D0        | 3736h | RXFBC    |
| 37F5h | RXB1DLC     | 37D5h | TXB1DLC     | 37B5h | RXF5SIDH | 3795h | B5DLC       | 3775h | B3DLC       | 3755h | B1DLC       | 3735h | RXFBC    |
| 37F4h | RXB1EIDL    | 37D4h | TXB1EIDL    | 37B4h | RXF5SIDH | 3794h | B5EIDL      | 3774h | B3EIDL      | 3754h | B1EIDL      | 3734h | RXFBC    |
| 37F3h | RXB1EIDL    | 37D3h | TXB1EIDL    | 37B3h | RXF4EIDL | 3793h | B5EIDL      | 3773h | B3EIDL      | 3753h | B1EIDL      | 3733h | RXFBC    |
| 37F2h | RXB1SIDL    | 37D2h | TXB1SIDL    | 37B2h | RXF4EIDL | 3792h | B5SIDL      | 3772h | B3SIDL      | 3752h | B1SIDL      | 3732h | RXFBC    |
| 37F1h | RXB1SIDH    | 37D1h | TXB1SIDH    | 37B1h | RXF4SIDL | 3791h | B5SIDH      | 3771h | B3SIDH      | 3751h | B1SIDH      | 3731h | RXFBC    |
| 37F0h | RXB1CON     | 37D0h | TXB1CON     | 37B0h | RXF4SIDH | 3790h | B5CON       | 3770h | B3CON       | 3750h | B1CON       | 3730h | SDF      |
| 37EFh | CANCON_RO1  | 37CFh | CANCON_RO3  | 37AFh | RXF3EIDL | 378Fh | CANCON_RO5  | 376Fh | CANCON_RO7  | 374Fh | CANCON_RO9  | 372Fh | RXF15    |
| 37EDh | CANSTAT_RO1 | 37CEh | CANSTAT_RO3 | 37AEh | RXF3EIDL | 378Eh | CANSTAT_RO5 | 376Eh | CANSTAT_RO7 | 374Eh | CANSTAT_RO9 | 372Eh | RXF15    |
| 37EDh | TXB0D7      | 37CDh | TXB2D7      | 37ADh | RXF3SIDL | 378Dh | B4D7        | 376Dh | B2D7        | 374Dh | B0D7        | 372Dh | RXF15    |
| 37ECh | TXB0D6      | 37CCh | TXB2D6      | 37ACH | RXF3SIDH | 378Ch | B4D6        | 376Ch | B2D6        | 374Ch | B0D6        | 372Ch | RXF15    |
| 37EBh | TXB0D5      | 37CBh | TXB2D5      | 37ABh | RXF2EIDL | 378Bh | B4D5        | 376Bh | B2D5        | 374Bh | B0D5        | 372Bh | RXF14    |
| 37EAh | TXB0D4      | 37CAh | TXB2D4      | 37AAh | RXF2EIDL | 378Ah | B4D4        | 376Ah | B2D4        | 374Ah | B0D4        | 372Ah | RXF14    |
| 37E9h | TXB0D3      | 37C9h | TXB2D3      | 37A9h | RXF2SIDL | 3789h | B4D3        | 3769h | B2D3        | 3749h | B0D3        | 3729h | RXF14    |
| 37E8h | TXB0D2      | 37C8h | TXB2D2      | 37A8h | RXF2SIDH | 3788h | B4D2        | 3768h | B2D2        | 3748h | B0D2        | 3728h | RXF14    |
| 37E7h | TXB0D1      | 37C7h | TXB2D1      | 37A7h | RXF1EIDL | 3787h | B4D1        | 3767h | B2D1        | 3747h | B0D1        | 3727h | RXF13    |
| 37E6h | TXB0D0      | 37C6h | TXB2D0      | 37A6h | RXF1EIDL | 3786h | B4D0        | 3766h | B2D0        | 3746h | B0D0        | 3726h | RXF13    |
| 37E5h | TXB0DLC     | 37C5h | TXB2DLC     | 37A5h | RXF1SIDL | 3785h | B4DLC       | 3765h | B2DLC       | 3745h | B0DLC       | 3725h | RXF13    |
| 37E4h | TXB0EIDL    | 37C4h | TXB2EIDL    | 37A4h | RXF1SIDH | 3784h | B4EIDL      | 3764h | B2EIDL      | 3744h | B0EIDL      | 3724h | RXF13    |
| 37E3h | TXB0EIDL    | 37C3h | TXB2EIDL    | 37A3h | RXF0EIDL | 3783h | B4EIDL      | 3763h | B2EIDL      | 3743h | B0EIDL      | 3723h | RXF12    |
| 37E2h | TXB0SIDL    | 37C2h | TXB2SIDL    | 37A2h | RXF0EIDL | 3782h | B4SIDL      | 3762h | B2SIDL      | 3742h | B0SIDL      | 3722h | RXF12    |
| 37E1h | TXB0SIDH    | 37C1h | TXB2SIDH    | 37A1h | RXF0SIDL | 3781h | B4SIDH      | 3761h | B2SIDH      | 3741h | B0SIDH      | 3721h | RXF12    |
| 37E0h | TXB0CON     | 37C0h | TXB2CON     | 37A0h | RXF0SIDH | 3780h | B4CON       | 3760h | B2CON       | 3740h | B0CON       | 3720h | RXF12    |

**Legend:** Unimplemented data memory locations and registers, read as '0'.

## 4.5.5 STATUS REGISTER

The STATUS register, shown in [Register 4-2](#), contains the arithmetic status of the ALU. As with any other SFR, it can be the operand for any instruction.

If the STATUS register is the destination for an instruction that affects the Z, DC, C, OV or N bits, the results of the instruction are not written; instead, the STATUS register is updated according to the instruction performed. Therefore, the result of an instruction with the STATUS register as its destination may be different than intended. As an example, `CLRF STATUS` will set the Z bit and leave the remaining Status bits unchanged ('0uuu u1uu').

It is recommended that only `BCF`, `BSF`, `SWAPF`, `MOVFF`, `MOVWF` and `MOVFFL` instructions are used to alter the STATUS register, because these instructions do not affect the Z, C, DC, OV or N bits in the STATUS register.

For other instructions that do not affect Status bits, see the instruction set summaries in [Section 42.2 “Extended Instruction Set”](#) and [Table 42-3](#).

**Note:** The C and DC bits operate as the borrow and digit borrow bits, respectively, in subtraction.

## 4.5.6 CALL SHADOW REGISTER

When `CALL`, `CALLW`, `RCALL` instructions are used, the WREG, BSR and STATUS are automatically saved in hardware and can be accessed using the WREG\_CSHAD, BSR\_CSHAD and STATUS\_CSHAD registers.

- Note 1:** Changing the values of these registers may lead to erroneous code execution.
- 2:** If the contents of these registers are not handled correctly, it may lead to erroneous code execution.

## 4.6 Register Definitions: Status Registers

### REGISTER 4-2: STATUS: STATUS REGISTER

| U-0   | R-1/q                  | R-1/q                  | R/W-0/u | R/W-0/u | R/W-0/u | R/W-0/u | R/W-0/u |
|-------|------------------------|------------------------|---------|---------|---------|---------|---------|
| —     | $\overline{\text{TO}}$ | $\overline{\text{PD}}$ | N       | OV      | Z       | DC      | C       |
| bit 7 |                        |                        |         |         |         |         | bit 0   |

#### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7 **Unimplemented:** Read as '0'

bit 6  **$\overline{\text{TO}}$ :** Time-Out bit

1 = Set at power-up or by execution of `CLRWDT` or `SLEEP` instruction

0 = A WDT time-out occurred

bit 5  **$\overline{\text{PD}}$ :** Power-Down bit

1 = Set at power-up or by execution of `CLRWDT` instruction

0 = Set by execution of the `SLEEP` instruction

bit 4 **N:** Negative bit used for signed arithmetic (2's complement); indicates if the result is negative, (ALU MSb = 1).

1 = The result is negative

0 = The result is positive

bit 3 **OV:** Overflow bit used for signed arithmetic (2's complement); indicates an overflow of the 7-bit magnitude, which causes the sign bit (bit 7) to change state.

1 = Overflow occurred for current signed arithmetic operation

0 = No overflow occurred

bit 2 **Z:** Zero bit

1 = The result of an arithmetic or logic operation is zero

0 = The result of an arithmetic or logic operation is not zero

bit 1 **DC:** Digit Carry/Borrow bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions)<sup>(1)</sup>

1 = A carry-out from the 4th low-order bit of the result occurred

0 = No carry-out from the 4th low-order bit of the result

bit 0 **C:** Carry/Borrow bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions)<sup>(1,2)</sup>

1 = A carry-out from the Most Significant bit of the result occurred

0 = No carry-out from the Most Significant bit of the result occurred

**Note 1:** For `Borrow`, the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand.

**2:** For Rotate (`RRF`, `RLF`) instructions, this bit is loaded with either the high or low-order bit of the Source register.



## 4.7 Data Addressing Modes

**Note:** The execution of some instructions in the core PIC18 instruction set are changed when the PIC18 extended instruction set is enabled. See [Section 4.8 “Data Memory and the Extended Instruction Set”](#) for more information.

While the program memory can be addressed in only one way – through the Program Counter – information in the data memory space can be addressed in several ways. For most instructions, the addressing mode is fixed. Other instructions may use up to three modes, depending on which operands are used and whether or not the extended instruction set is enabled.

The addressing modes are:

- Inherent
- Literal
- Direct
- Indirect

An additional addressing mode, Indexed Literal Offset, is available when the extended instruction set is enabled (XINST Configuration bit = 1). Its operation is discussed in detail in [Section 4.8.1 “Indexed Addressing with Literal Offset”](#).

### 4.7.1 INHERENT AND LITERAL ADDRESSING

Many PIC18 control instructions do not need any argument at all; they either perform an operation that globally affects the device or they operate implicitly on one register. This addressing mode is known as Inherent Addressing. Examples include `SLEEP`, `RESET` and `DAW`.

Other instructions work in a similar way but require an additional explicit argument in the opcode. This is known as Literal Addressing mode because they require some literal value as an argument. Examples include `ADDLW` and `MOVLW`, which respectively, add or move a literal value to the W register. Other examples include `CALL` and `GOTO`, which include a 20-bit program memory address.

## 4.7.2 DIRECT ADDRESSING

Direct addressing specifies all or part of the source and/or destination address of the operation within the opcode itself. The options are specified by the arguments accompanying the instruction.

In the core PIC18 instruction set, bit-oriented and byte-oriented instructions use some version of direct addressing by default. All of these instructions include some 8-bit literal address as their Least Significant Byte. This address specifies either a register address in one of the banks of data RAM ([Section 4.5.2 “General Purpose Register File”](#)) or a location in the Access Bank ([Section 4.5.4 “Access Bank”](#)) as the data source for the instruction.

The Access RAM bit ‘a’ determines how the address is interpreted. When ‘a’ is ‘1’, the contents of the BSR ([Section 4.5.1 “Bank Select Register \(BSR\)”](#)) are used with the address to determine the complete 14-bit address of the register. When ‘a’ is ‘0’, the address is interpreted as being a register in the Access Bank. Addressing that uses the Access RAM is sometimes also known as Direct Forced Addressing mode.

A few instructions, such as `MOVFFL`, include the entire 14-bit address (either source or destination) in their opcodes. In these cases, the BSR is ignored entirely.

The destination of the operation’s results is determined by the destination bit ‘d’. When ‘d’ is ‘1’, the results are stored back in the source register, overwriting its original contents. When ‘d’ is ‘0’, the results are stored in the W register. Instructions without the ‘d’ argument have a destination that is implicit in the instruction; their destination is either the target register being operated on or the W register.

## 4.7.3 INDIRECT ADDRESSING

Indirect addressing allows the user to access a location in data memory without giving a fixed address in the instruction. This is done by using File Select Registers (FSRs) as pointers to the locations which are to be read or written. Since the FSRs are themselves located in RAM as Special File Registers, they can also be directly manipulated under program control. This makes FSRs very useful in implementing data structures, such as tables and arrays in data memory.

The registers for indirect addressing are also implemented with Indirect File Operands (INDFs) that permit automatic manipulation of the pointer value with auto-incrementing, auto-decrementing or offsetting with another value. This allows for efficient code, using loops, such as the example of clearing an entire RAM bank in [Example 4-6](#).

### EXAMPLE 4-6: HOW TO CLEAR RAM (BANK 1) USING INDIRECT ADDRESSING

|          |       |              |                 |
|----------|-------|--------------|-----------------|
|          | LFSR  | FSR0, 100h ; |                 |
| NEXT     | CLRF  | POSTINC0 ;   | Clear INDF      |
|          |       |              | ; register then |
|          |       |              | ; inc pointer   |
|          | BTFSS | FSR0H, 1 ;   | All done with   |
|          |       |              | ; Bank1?        |
|          | BRA   | NEXT ;       | NO, clear next  |
| CONTINUE |       |              | ; YES, continue |

#### 4.7.3.1 FSR Registers and the INDF Operand

At the core of indirect addressing are three sets of registers: FSR0, FSR1 and FSR2. Each represents a pair of 8-bit registers, FSRnH and FSRnL. Each FSR pair holds a 14-bit value, therefore, the two upper bits of the FSRnH register are not used. The 14-bit FSR value can address the entire range of the data memory in a linear fashion. The FSR register pairs, then, serve as pointers to data memory locations.

Indirect addressing is accomplished with a set of Indirect File Operands, INDF0 through INDF2. These can be thought of as “virtual” registers; they are mapped in the SFR space but are not physically implemented. Reading or writing to a particular INDF register actually accesses the data addressed by its corresponding FSR register pair. A read from INDF1, for example, reads the data at the address indicated by FSR1H:FSR1L. Instructions that use the INDF registers as operands actually use the contents of their corresponding FSR as a pointer to the instruction’s target. The INDF operand is just a convenient way of using the pointer.

Because indirect addressing uses a full 14-bit address, data RAM banking is not necessary. Thus, the current contents of the BSR and the Access RAM bit have no effect on determining the target address.

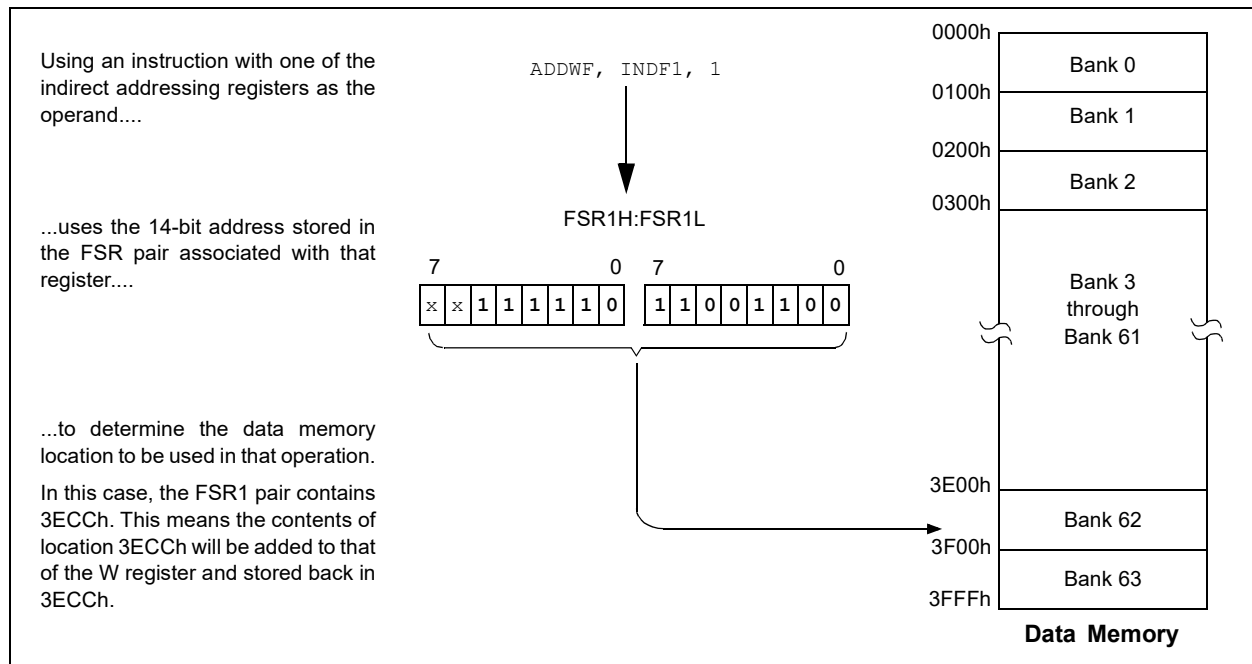
## 4.7.3.2 FSR Registers, POSTINC, POSTDEC, PREINC and PLUSW

In addition to the INDF operand, each FSR register pair also has four additional indirect operands. Like INDF, these are “virtual” registers which cannot be directly read or written. Accessing these registers actually accesses the location to which the associated FSR register pair points, and also performs a specific action on the FSR value. They are:

- **POSTDEC:** accesses the location to which the FSR points, then automatically decrements the FSR by 1 afterwards
- **POSTINC:** accesses the location to which the FSR points, then automatically increments the FSR by 1 afterwards
- **PREINC:** automatically increments the FSR by 1, then uses the location to which the FSR points in the operation
- **PLUSW:** adds the signed value of the W register (range of -127 to 128) to that of the FSR and uses the location to which the result points in the operation.

In this context, accessing an INDF register uses the value in the associated FSR register without changing it. Similarly, accessing a PLUSW register gives the FSR value an offset by that in the W register; however, neither W nor the FSR is actually changed in the operation. Accessing the other virtual registers changes the value of the FSR register.

**FIGURE 4-6: INDIRECT ADDRESSING**



Operations on the FSRs with POSTDEC, POSTINC and PREINC affect the entire register pair; that is, roll-overs of the FSRnL register from FFh to 00h carry over to the FSRnH register. On the other hand, results of these operations do not change the value of any flags in the STATUS register (e.g., Z, N, OV, etc.).

The PLUSW register can be used to implement a form of indexed addressing in the data memory space. By manipulating the value in the W register, users can reach addresses that are fixed offsets from pointer addresses. In some applications, this can be used to implement some powerful program control structure, such as software stacks, inside of data memory.

#### 4.7.3.3 Operations by FSRs on FSRs

Indirect addressing operations that target other FSRs or virtual registers represent special cases. For example, using an FSR to point to one of the virtual registers will not result in successful operations. As a specific case, assume that FSR0H:FSR0L contains 3FE7h, the address of INDF1. Attempts to read the value of the INDF1 using INDF0 as an operand will return 00h. Attempts to write to INDF1 using INDF0 as the operand will result in a NOP.

On the other hand, using the virtual registers to write to an FSR pair may not occur as planned. In these cases, the value will be written to the FSR pair but without any incrementing or decrementing. Thus, writing to either the INDF2 or POSTDEC2 register will write the same value to the FSR2H:FSR2L.

Since the FSRs are physical registers mapped in the SFR space, they can be manipulated through all direct operations. Users should proceed cautiously when working on these registers, particularly if their code uses indirect addressing.

Similarly, operations by indirect addressing are generally permitted on all other SFRs. Users should exercise the appropriate caution that they do not inadvertently change settings that might affect the operation of the device.

## 4.8 Data Memory and the Extended Instruction Set

Enabling the PIC18 extended instruction set (XINST Configuration bit = 1) significantly changes certain aspects of data memory and its addressing. Specifically, the use of the Access Bank for many of the core PIC18 instructions is different; this is due to the introduction of a new addressing mode for the data memory space.

What does not change is just as important. The size of the data memory space is unchanged, as well as its linear addressing. The SFR map remains the same. Core PIC18 instructions can still operate in both Direct and Indirect Addressing mode; inherent and literal instructions do not change at all. Indirect addressing with FSR0 and FSR1 also remain unchanged.

### 4.8.1 INDEXED ADDRESSING WITH LITERAL OFFSET

Enabling the PIC18 extended instruction set changes the behavior of indirect addressing using the FSR2 register pair within Access RAM. Under the proper conditions, instructions that use the Access Bank – that is, most bit-oriented and byte-oriented instructions – can invoke a form of indexed addressing using an offset specified in the instruction. This special addressing mode is known as Indexed Addressing with Literal Offset, or Indexed Literal Offset mode.

When using the extended instruction set, this addressing mode requires the following:

- The use of the Access Bank is forced ('a' = 0) and
- The file address argument is less than or equal to 5Fh.

Under these conditions, the file address of the instruction is not interpreted as the lower byte of an address (used with the BSR in direct addressing), or as an 8-bit address in the Access Bank. Instead, the value is interpreted as an offset value to an Address Pointer, specified by FSR2. The offset and the contents of FSR2 are added to obtain the target address of the operation.

### 4.8.2 INSTRUCTIONS AFFECTED BY INDEXED LITERAL OFFSET MODE

Any of the core PIC18 instructions that can use direct addressing are potentially affected by the Indexed Literal Offset Addressing mode. This includes all byte-oriented and bit-oriented instructions, or almost one-half of the standard PIC18 instruction set. Instructions that only use Inherent or Literal Addressing modes are unaffected.

Additionally, byte-oriented and bit-oriented instructions are not affected if they do not use the Access Bank (Access RAM bit is '1'), or include a file address of 60h or above. Instructions meeting these criteria will continue to execute as before. A comparison of the different possible addressing modes when the extended instruction set is enabled is shown in [Figure 4-7](#).

Those who desire to use byte-oriented or bit-oriented instructions in the Indexed Literal Offset mode should note the changes to assembler syntax for this mode. This is described in more detail in [Section 42.2.1 “Extended Instruction Syntax”](#).

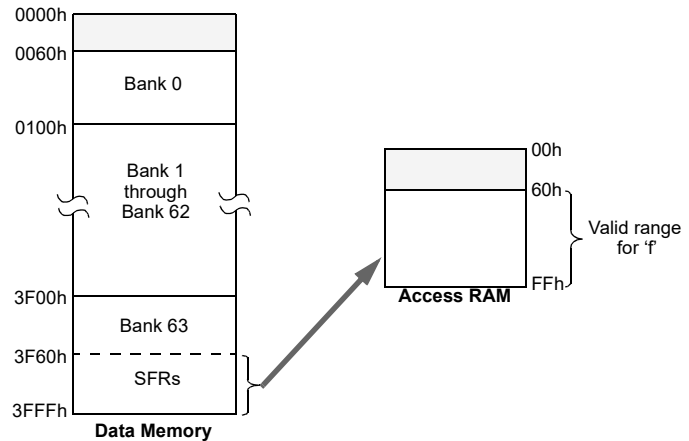
**FIGURE 4-7: COMPARING ADDRESSING OPTIONS FOR BIT-ORIENTED AND BYTE-ORIENTED INSTRUCTIONS (EXTENDED INSTRUCTION SET ENABLED)**

**EXAMPLE INSTRUCTION:** `ADDWF, f, d, a` (Opcode: `0010 01da ffff ffff`)

**When 'a' = 0 and  $f \geq 60h$ :**

The instruction executes in Direct Forced mode. 'f' is interpreted as a location in the Access RAM between 060h and 0FFh. This is the same as locations 3F60h to 3FFFh (Bank 63) of data memory.

Locations below 60h are not available in this Addressing mode.



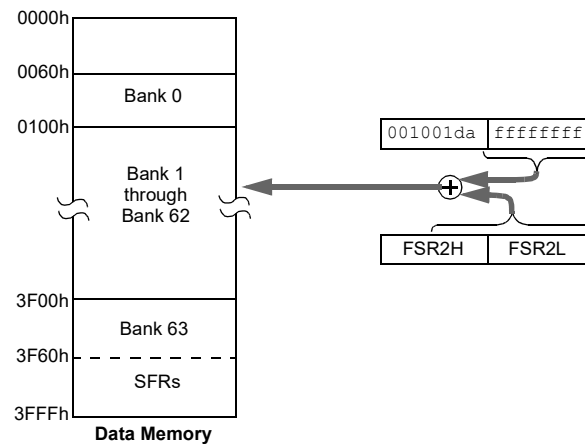
**When 'a' = 0 and  $f \leq 5Fh$ :**

The instruction executes in Indexed Literal Offset mode. 'f' is interpreted as an offset to the address value in FSR2. The two are added together to obtain the address of the target register for the instruction. The address can be anywhere in the data memory space.

Note that in this mode, the correct syntax is now:

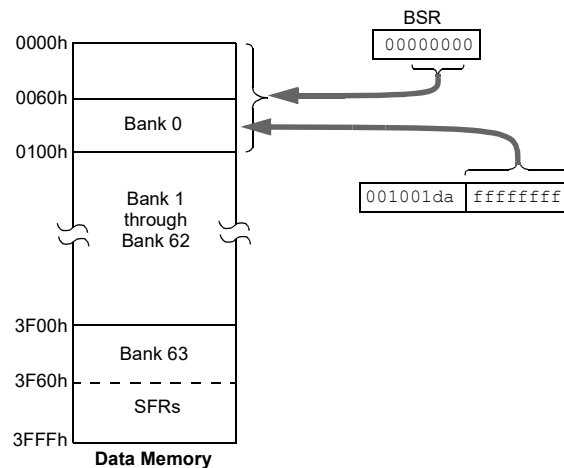
`ADDWF [k], d`

where 'k' is the same as 'f'.



**When 'a' = 1 (all values of f):**

The instruction executes in Direct mode (also known as Direct Long mode). 'f' is interpreted as a location in one of the 63 banks of the data memory space. The bank is designated by the Bank Select Register (BSR). The address can be in any implemented bank in the data memory space.



## 4.8.3 MAPPING THE ACCESS BANK IN INDEXED LITERAL OFFSET MODE

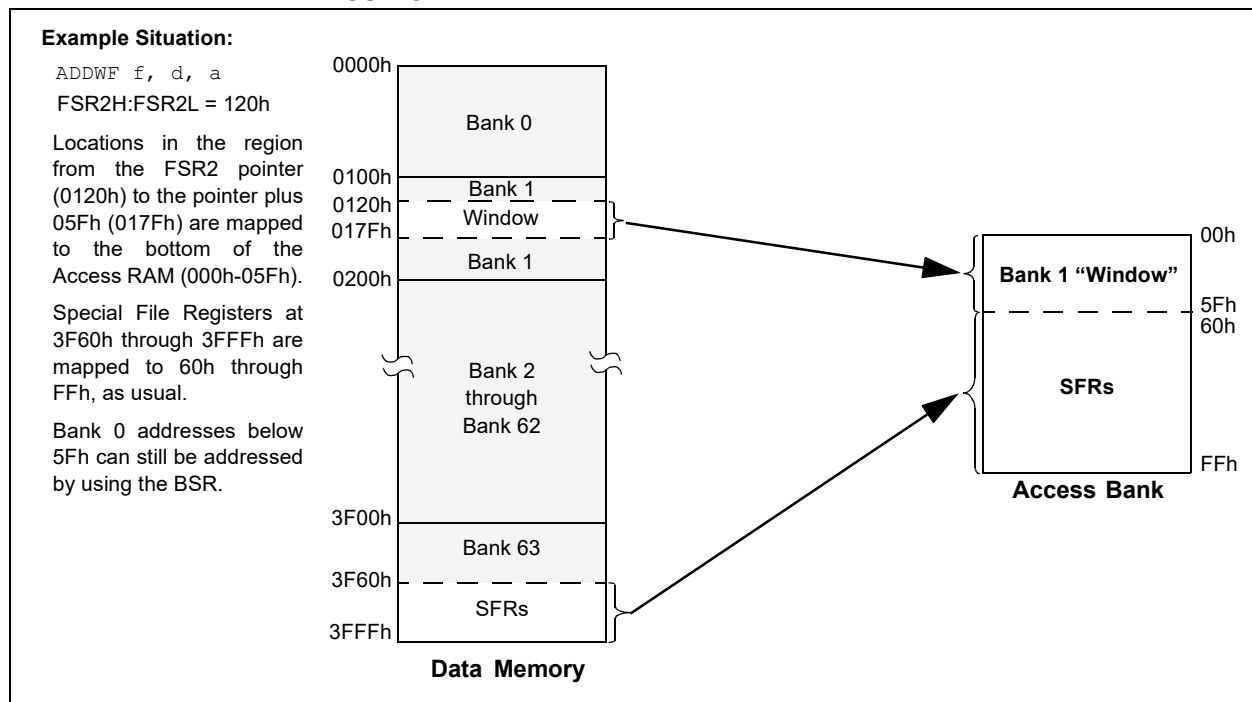
The use of Indexed Literal Offset Addressing mode effectively changes how the first 96 locations of Access RAM (00h to 5Fh) are mapped. Rather than containing just the contents of the bottom section of Bank 0, this mode maps the contents from a user defined “window” that can be located anywhere in the data memory space. The value of FSR2 establishes the lower boundary of the addresses mapped into the window, while the upper boundary is defined by FSR2 plus 95 (5Fh). Addresses in the Access RAM above 5Fh are mapped as previously described (see [Section 4.5.4 “Access Bank”](#)). An example of Access Bank remapping in this addressing mode is shown in [Figure 4-8](#).

Remapping of the Access Bank applies *only* to operations using the Indexed Literal Offset mode. Operations that use the BSR (Access RAM bit is ‘1’) will continue to use direct addressing as before.

## 4.9 PIC18 Instruction Execution and the Extended Instruction Set

Enabling the extended instruction set adds eight additional commands to the existing PIC18 instruction set. These instructions are executed as described in [Section 42.2 “Extended Instruction Set”](#).

**FIGURE 4-8: REMAPPING THE ACCESS BANK WITH INDEXED LITERAL OFFSET ADDRESSING**



## 5.0 DEVICE CONFIGURATION

Device configuration consists of the Configuration Words, User ID, Device ID, Rev ID, Device Information Area (DIA), (see [Section 5.7 “Device Information Area”](#)), and the Device Configuration Information (DCI) regions, (see [Section 5.8 “Device Configuration Information”](#)).

### 5.1 Configuration Words

There are six Configuration Word bits that allow the user to setup the device with several choices of oscillators, Resets and memory protection options. These are implemented as Configuration Word 1 through Configuration Word 6 at 300000h through 300008h.

## 5.2 Register Definitions: Configuration Words

### REGISTER 5-1: CONFIGURATION WORD 1L (30 0000h)

| U-1   | R/W-1       | R/W-1 | R/W-1 | U-1 | R/W-1        | R/W-1 | R/W-1 |
|-------|-------------|-------|-------|-----|--------------|-------|-------|
| —     | RSTOSC<2:0> |       |       | —   | FEXTOSC<2:0> |       |       |
| bit 7 |             |       |       |     |              |       | bit 0 |

#### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '1'

-n = Value for blank device

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7 **Unimplemented:** Read as '1'

bit 6-4 **RSTOSC<2:0>:** Power-up Default Value for COSC bits

111 = EXTOSC operating per FEXTOSC<2:0> bits

110 = HFINTOSC with HFFRQ = 4 MHz and CDIV = 4:1

101 = LFINTOSC

100 = SOSC

011 = Reserved

010 = EXTOSC with 4x PLL, with EXTOSC operating per FEXTOSC<2:0> bits

001 = Reserved

000 = HFINTOSC with HFFRQ = 64 MHz and CDIV = 1:1; resets COSC/NOSC to 3'b110

bit 3 **Unimplemented:** Read as '1'

bit 2-0 **FEXTOSC<2:0>:** FEXTOSC External Oscillator Mode Selection bits

111 = ECH (External Clock High Power)

110 = ECM (External Clock Medium Power)

101 = ECL (External Clock Low Power)

100 = Oscillator is not enabled

011 = Reserved (do not use)

010 = HS (crystal oscillator) above 4 MHz

001 = XT (crystal oscillator) above 100 kHz, below 4 MHz

000 = LP (crystal oscillator) optimized for 32.768 kHz



## REGISTER 5-2: CONFIGURATION WORD 1H (30 0001h)

| U-1   | U-1 | R/W-1 | U-1 | R/W-1 | U-1 | R/W-1  | R/W-1    |
|-------|-----|-------|-----|-------|-----|--------|----------|
| —     | —   | FCMEN | —   | CSWEN | —   | PR1WAY | CLKOUTEN |
| bit 7 |     |       |     |       |     |        | bit 0    |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '1'

-n = Value for blank device

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-6 **Unimplemented:** Read as '1'

bit 5 **FCMEN:** Fail-Safe Clock Monitor Enable bit

1 = FSCM timer is enabled

0 = FSCM timer is disabled

bit 4 **Unimplemented:** Read as '1'

bit 3 **CSWEN:** Clock Switch Enable bit

1 = Writing to NOSC and NDIV is allowed

0 = The NOSC and NDIV bits cannot be changed by user software

bit 2 **Unimplemented:** Read as '1'

bit 1 **PR1WAY:** PRLOCKED One-Way Set Enable bit

1 = PRLOCKED bit can be cleared and set only once; Priority registers remain locked after one clear/set cycle

0 = PRLOCKED bit can be set and cleared multiple times (subject to the unlock sequence)

bit 0 **CLKOUTEN:** Clock Out Enable bit

If FEXTOSC<2:0> = EC (high, mid or low) or Not Enabled:

1 = CLKOUT function is disabled; I/O or oscillator function on OSC2

0 = CLKOUT function is enabled; Fosc/4 clock appears at OSC2

Otherwise:

This bit is ignored.

## REGISTER 5-3: CONFIGURATION WORD 2L (30 0002h)

| R/W-1      | R/W-1 | R/W-1                              | R/W-1   | R/W-1  | R/W-1      | R/W-1 | R/W-1 |
|------------|-------|------------------------------------|---------|--------|------------|-------|-------|
| BOREN<1:0> |       | $\overline{\text{LPBOR}}\text{EN}$ | IVT1WAY | MVECEN | PWRTS<1:0> |       | MCLRE |
| bit 7      |       |                                    |         |        |            |       | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '1'

-n = Value for blank device

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-6

**BOREN<1:0>**: Brown-out Reset Enable bits

When enabled, Brown-out Reset Voltage (VBOR) is set by the BORV bit.

11 = Brown-out Reset is enabled, SBOREN bit is ignored

10 = Brown-out Reset is enabled while running, disabled in Sleep; SBOREN is ignored

01 = Brown-out Reset is enabled according to SBOREN

00 = Brown-out Reset is disabled

bit 5

**LPBOR**EN: Low-Power BOR Enable bit

1 = Low-Power BOR is disabled

0 = Low-Power BOR is enabled

bit 4

**IVT1WAY**: IVTLOCK bit One-Way Set Enable bit

1 = IVTLOCKED bit can be cleared and set only once; IVT registers remain locked after one clear/set cycle

0 = IVTLOCK ED bit can be set and cleared multiple times (subject to the unlock sequence)

bit 3

**MVECEN**: Multi-vector Enable bit

1 = Multi-vector enabled; Vector table used for interrupts

0 = Legacy interrupt behavior

bit 2-1

**PWRTS<1:0>**: Power-up Timer Selection bits

11 = PWRT is disabled

10 = PWRT set at 64 ms

01 = PWRT set at 16 ms

00 = PWRT set at 1 ms

bit 0

**MCLRE**: Master Clear ( $\overline{\text{MCLR}}$ ) Enable bit

If LVP = 1:

RE3 pin function is  $\overline{\text{MCLR}}$

If LVP = 0:

1 =  $\overline{\text{MCLR}}$  pin is  $\overline{\text{MCLR}}$

0 =  $\overline{\text{MCLR}}$  pin function is a port defined function

## REGISTER 5-4: CONFIGURATION WORD 2H (30 0003h)

| R/W-1                     | U-1 | R/W-1                     | R/W-1  | R/W-1   | R/W-1                   | R/W-1                    | R/W-1 |
|---------------------------|-----|---------------------------|--------|---------|-------------------------|--------------------------|-------|
| $\overline{\text{XINST}}$ | —   | $\overline{\text{DEBUG}}$ | STVREN | PPS1WAY | $\overline{\text{ZCD}}$ | BORV<1:0> <sup>(1)</sup> |       |
| bit 7                     |     |                           |        |         |                         |                          | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '1'

-n = Value for blank device

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7  **$\overline{\text{XINST}}$** : Extended Instruction Set Enable bit

1 = Extended instruction set and Indexed Addressing mode are disabled (Legacy mode)

0 = Extended instruction set and Indexed Addressing mode are enabled

bit 6 **Unimplemented**: Read as '1'

bit 5  **$\overline{\text{DEBUG}}$** : Debugger Enable bit

1 = Background debugger is disabled

0 = Background debugger is enabled

bit 4 **STVREN**: Stack Overflow/Underflow Reset Enable bit

1 = Stack Overflow or Underflow will cause a Reset

0 = Stack Overflow or Underflow will not cause a Reset

bit 3 **PPS1WAY**: PPSLOCKED One-Way Set Enable bit

1 = PPSLOCKED bit can be cleared and set only once; PPS registers remain locked after one clear/set cycle

0 = PPSLOCKED bit can be set and cleared multiple times (subject to the unlock sequence)

bit 2  **$\overline{\text{ZCD}}$** : Zero-Cross Detect Enable bit

1 = ZCD is disabled; ZCD can be enabled by setting the bit SEN of the ZCDCON register

0 = ZCD is always enabled

bit 1-0 **BORV<1:0>**: Brown-out Reset Voltage Selection bits<sup>(1)</sup>

PIC18F25/26K83 Devices:

11 = Brown-out Reset Voltage (VBOR) is set to 2.45V

10 = Brown-out Reset Voltage (VBOR) is set to 2.45V

01 = Brown-out Reset Voltage (VBOR) is set to 2.7V

00 = Brown-out Reset Voltage (VBOR) is set to 2.85V

PIC18LF25/26K83 Device:

11 = Brown-out Reset Voltage (VBOR) is set to 1.90V

10 = Brown-out Reset Voltage (VBOR) is set to 2.45V

01 = Brown-out Reset Voltage (VBOR) is set to 2.7V

00 = Brown-out Reset Voltage (VBOR) is set to 2.85V

**Note 1:** The higher voltage setting is recommended for operation at or above 16 MHz.

# PIC18(L)F25/26K83

## REGISTER 5-5: CONFIGURATION WORD 3L (30 0004h)

|       |           |       |             |       |       |       |       |
|-------|-----------|-------|-------------|-------|-------|-------|-------|
| U-1   | R/W-1     | R/W-1 | R/W-1       | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
| —     | WDTE<1:0> |       | WDTCPs<4:0> |       |       |       |       |
| bit 7 |           |       |             |       |       |       | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '1'

-n = Value for blank device

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7

**Unimplemented:** Read as '1'

bit 6-5

**WDTE<1:0>:** WDT Operating Mode bits

00 = WDT is disabled, SWDTEN is ignored

01 = WDT is enabled/disabled by the SWDTEN bit in WDTCON0

10 = WDT is enabled while Sleep = 0, suspended when Sleep = 1; SWDTEN is ignored

11 = WDT is enabled regardless of Sleep; SWDTEN is ignored

bit 4-0

**WDTCPs<4:0>:** WDT Period Select bits

| WDTCPs<4:0> | WDTPS at POR |               |                 |                                             | Software Control of WDTPS? |
|-------------|--------------|---------------|-----------------|---------------------------------------------|----------------------------|
|             | Value        | Divider Ratio |                 | Typical Time-out (F <sub>IN</sub> = 31 kHz) |                            |
| 00000       | 00000        | 1:32          | 2 <sup>5</sup>  | 1 ms                                        | No                         |
| 00001       | 00001        | 1:64          | 2 <sup>6</sup>  | 2 ms                                        |                            |
| 00010       | 00010        | 1:128         | 2 <sup>7</sup>  | 4 ms                                        |                            |
| 00011       | 00011        | 1:256         | 2 <sup>8</sup>  | 8 ms                                        |                            |
| 00100       | 00100        | 1:512         | 2 <sup>9</sup>  | 16 ms                                       |                            |
| 00101       | 00101        | 1:1024        | 2 <sup>10</sup> | 32 ms                                       |                            |
| 00110       | 00110        | 1:2048        | 2 <sup>11</sup> | 64 ms                                       |                            |
| 00111       | 00111        | 1:4096        | 2 <sup>12</sup> | 128 ms                                      |                            |
| 01000       | 01000        | 1:8192        | 2 <sup>13</sup> | 256 ms                                      |                            |
| 01001       | 01001        | 1:16384       | 2 <sup>14</sup> | 512 ms                                      |                            |
| 01010       | 01010        | 1:32768       | 2 <sup>15</sup> | 1s                                          |                            |
| 01011       | 01011        | 1:65536       | 2 <sup>16</sup> | 2s                                          |                            |
| 01100       | 01100        | 1:131072      | 2 <sup>17</sup> | 4s                                          |                            |
| 01101       | 01101        | 1:262144      | 2 <sup>18</sup> | 8s                                          |                            |
| 01110       | 01110        | 1:524299      | 2 <sup>19</sup> | 16s                                         |                            |
| 01111       | 01111        | 1:1048576     | 2 <sup>20</sup> | 32s                                         |                            |
| 10000       | 10000        | 1:2097152     | 2 <sup>21</sup> | 64s                                         |                            |
| 10001       | 10001        | 1:4194304     | 2 <sup>22</sup> | 128s                                        |                            |
| 10010       | 10010        | 1:8388608     | 2 <sup>23</sup> | 256s                                        |                            |
| 10011       | 10011        | 1:32          | 2 <sup>5</sup>  | 1 ms                                        | No                         |
| ...         | ...          |               |                 |                                             |                            |
| 11110       | 11110        | 1:65536       | 2 <sup>16</sup> | 2s                                          | Yes                        |
| 11111       | 01011        |               |                 |                                             |                            |

# PIC18(L)F25/26K83

## REGISTER 5-6: CONFIGURATION WORD 3H (30 0005h)

|       |     |             |       |       |             |       |       |
|-------|-----|-------------|-------|-------|-------------|-------|-------|
| U-1   | U-1 | R/W-1       | R/W-1 | R/W-1 | R/W-1       | R/W-1 | R/W-1 |
| —     | —   | WDTCCS<2:0> |       |       | WDTCWS<2:0> |       |       |
| bit 7 |     |             |       |       |             |       | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '1'

-n = Value for blank device

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-6

**Unimplemented:** Read as '1'

bit 5-3

**WDTCCS<2:0>:** WDT Input Clock Selector bits

If WDTE<1:0> Fuses = 2'b00:

These bits are ignored.

Otherwise:

000 = WDT reference clock is the 31.0 kHz LFINTOSC

001 = WDT reference clock is the 31.25 kHz MFINTOSC

010 = WDT reference clock is SOSC

011 = Reserved (default to LFINTOSC)

•

•

110 = Reserved (default to LFINTOSC)

111 = Software control

bit 2-0

**WDTCWS<2:0>:** WDT Window Select bits

| WDTCWS<2:0> | Window at POR |                              |                                | Software Control of Window | Keyed Access Required? |
|-------------|---------------|------------------------------|--------------------------------|----------------------------|------------------------|
|             | Value         | Window Delay Percent of Time | Window Opening Percent of Time |                            |                        |
| 000         | 000           | 87.5                         | 12.5                           | No                         | Yes                    |
| 001         | 001           | 75                           | 25                             |                            |                        |
| 010         | 010           | 62.5                         | 37.5                           |                            |                        |
| 011         | 011           | 50                           | 50                             |                            |                        |
| 100         | 100           | 37.5                         | 62.5                           |                            |                        |
| 101         | 101           | 25                           | 75                             |                            |                        |
| 110         | 111           | n/a                          | 100                            |                            |                        |
| 111         | 111           | n/a                          | 100                            | Yes                        | No                     |

# PIC18(L)F25/26K83

## REGISTER 5-7: CONFIGURATION WORD 4L (30 0006h)

| R/W-1                 | U-1 | U-1 | R/W-1                | R/W-1               | R/W-1                      | R/W-1 | R/W-1 |
|-----------------------|-----|-----|----------------------|---------------------|----------------------------|-------|-------|
| WRTAPP <sup>(1)</sup> | —   | —   | SAFEN <sup>(1)</sup> | BBEN <sup>(1)</sup> | BBSIZE<2:0> <sup>(2)</sup> |       |       |
| bit 7                 |     |     |                      |                     |                            | bit 0 |       |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '1'

-n = Value for blank device

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

- bit 7 **WRTAPP:** Application Block Write Protection bit<sup>(1)</sup>  
 1 = Application Block is NOT write-protected  
 0 = Application Block is write-protected
- bit 6-5 **Unimplemented:** Read as '1'
- bit 4 **SAFEN:** Storage Area Flash Enable bit<sup>(1)</sup>  
 1 = SAF is disabled  
 0 = SAF is enabled
- bit 3 **BBEN:** Boot Block Enable bit<sup>(1)</sup>  
 1 = Boot Block disabled  
 0 = Boot Block enabled
- bit 2-0 **BBSIZE<2:0>:** Boot Block Size Selection bits<sup>(2)</sup>  
 Refer to [Table 5-1](#).

**Note 1:** Bits are implemented as sticky bits. Once protection is enabled through ICSP™ or a self-write, it can only be reset through a Bulk Erase.

**2:** BBSIZE<2:0> bits can only be changed when BBEN = 1. Once BBEN = 0, BBSIZE<2:0> can only be changed through a Bulk Erase.

**TABLE 5-1: BOOT BLOCK SIZE BITS**

| BBEN | BBSIZE[2:0] | Boot Block Size (words) | END_ADDRESS_BOOT | Device Size <sup>(1)</sup> |     |
|------|-------------|-------------------------|------------------|----------------------------|-----|
|      |             |                         |                  | 16k                        | 32k |
| 1    | xxx         | 0                       | —                | X                          | X   |
| 0    | 111         | 512                     | 00 03FFh         | X                          | X   |
| 0    | 110         | 1024                    | 00 07FFh         | X                          | X   |
| 0    | 101         | 2048                    | 00 0FFFh         | X                          | X   |
| 0    | 100         | 4096                    | 00 1FFFh         | X                          | X   |
| 0    | 011         | 8192                    | 00 3FFFh         | X                          | X   |
| 0    | 010         | 16384                   | 00 7FFFh         | —                          | X   |
| 0    | 001         | 32768                   | 00 FFFFh         | Note 2                     |     |
| 0    | 000         | 32768                   | 00 FFFFh         |                            |     |

**Note 1:** For each device, the quoted device size specification is listed in [Table 4-1](#).

**2:** The maximum Boot Block size is half the user program memory size. All selections higher than the maximum size default to maximum Boot Block size of half PFM. For example, all settings of BBSIZE = 000 through BBSIZE = 010, default to a Boot Block size of 16 kW on a 32 kW device.

## REGISTER 5-8: CONFIGURATION WORD 4H (30 0007h)

| U-1   | U-1 | R/W-1              | U-1 | R/W-1                   | R/W-1                 | R/W-1               | R/W-1                 |
|-------|-----|--------------------|-----|-------------------------|-----------------------|---------------------|-----------------------|
| —     | —   | LVP <sup>(2)</sup> | —   | WRTSAF <sup>(1,3)</sup> | WRTD <sup>(1,4)</sup> | WRTC <sup>(1)</sup> | WRTB <sup>(1,5)</sup> |
| bit 7 |     |                    |     | bit 0                   |                       |                     |                       |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '1'

-n = Value for blank device

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-6 **Unimplemented:** Read as '1'

bit 5 **LVP:** Low-Voltage Programming Enable bit<sup>(2)</sup>

1 = Low-voltage programming enabled. MCLR/VPP pin function is MCLR. MCLRE (Register 5-3) is ignored.

0 = HV on MCLR/VPP must be used for programming.

bit 4 **Unimplemented:** Read as '1'

bit 3 **WRTSAF:** Storage Area Flash (SAF) Write Protection bit<sup>(1,3)</sup>

1 = SAF is NOT write-protected

0 = SAF is write-protected

bit 2 **WRTD:** Data EEPROM Write Protection bit<sup>(1,4)</sup>

1 = Data EEPROM NOT write-protected

0 = Data EEPROM write-protected

bit 1 **WRTC:** Configuration Register Write Protection bit<sup>(1)</sup>

1 = Configuration Register NOT write-protected

0 = Configuration Register write-protected

bit 0 **WRTB:** Boot Block Write Protection bit<sup>(1,5)</sup>

1 = Boot Block NOT write-protected

0 = Boot Block write-protected

**Note 1:** Bits are implemented as sticky bits. Once protection is enabled through ICSP or a self write, it can only be reset through a Bulk Erase.

**2:** The LVP bit cannot be written (to zero) while operating from the LVP programming interface. The purpose of this rule is to prevent the user from dropping out of LVP mode while programming from LVP mode, or accidentally eliminating LVP mode from the configuration state.

**3:** Unimplemented if SAF is not present and only applicable if  $\overline{\text{SAFEN}} = 0$ .

**4:** Unimplemented if data EEPROM is not present.

**5:** Only applicable if  $\overline{\text{BBEN}} = 0$ .

# PIC18(L)F25/26K83

## REGISTER 5-9: CONFIGURATION WORD 5L (30 0008h)

|       |     |     |     |     |     |     |                        |
|-------|-----|-----|-----|-----|-----|-----|------------------------|
| U-1   | U-1 | U-1 | U-1 | U-1 | U-1 | U-1 | R/W-1                  |
| —     | —   | —   | —   | —   | —   | —   | $\overline{\text{CP}}$ |
| bit 7 |     |     |     |     |     |     | bit 0                  |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '1'

-n = Value for blank device

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-1 **Unimplemented:** Read as '1'

bit 0  **$\overline{\text{CP}}$ :** User Program Flash Memory and Data EEPROM Code Protection bit

1 = User Program Flash Memory and Data EEPROM code protection is disabled

0 = User Program Flash Memory and Data EEPROM code protection is enabled

## REGISTER 5-10: CONFIGURATION WORD 5H (30 0009h)

|       |     |     |     |     |     |     |       |
|-------|-----|-----|-----|-----|-----|-----|-------|
| U-1   | U-1 | U-1 | U-1 | U-1 | U-1 | U-1 | U-1   |
| —     | —   | —   | —   | —   | —   | —   | —     |
| bit 7 |     |     |     |     |     |     | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '1'

-n = Value for blank device

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-0 **Unimplemented:** Read as '1'

## TABLE 5-2: SUMMARY OF CONFIGURATION WORDS

| Address  | Name     | Bit 7      | Bit 6       | Bit 5       | Bit 4        | Bit 3   | Bit 2        | Bit 1     | Bit 0    | Default/<br>Unprogrammed<br>Value |
|----------|----------|------------|-------------|-------------|--------------|---------|--------------|-----------|----------|-----------------------------------|
| 30 0000h | CONFIG1L | —          | RSTOSC<2:0> |             |              | —       | FEXTOSC<2:0> |           |          | 1111 1111                         |
| 30 0001h | CONFIG1H | —          | —           | FCMEN       | —            | CSWEN   | —            | PR1WAY    | CLKOUTEN | 1111 1111                         |
| 30 0002h | CONFIG2L | BOREN<1:0> |             | LPBOREN     | IVT1WAY      | MVECEN  | PWRTS<1:0>   |           | MCLRE    | 1111 1111                         |
| 30 0003h | CONFIG2H | XINST      | —           | DEBUG       | STVREN       | PPS1WAY | ZCD          | BORV<1:0> |          | 1111 1111                         |
| 30 0004h | CONFIG3L | —          | WDTE<1:0>   |             | WDTCPSS<4:0> |         |              |           |          | 1111 1111                         |
| 30 0005h | CONFIG3H | —          | —           | WDTCCS<2:0> |              |         | WDTCCS<2:0>  |           |          | 1111 1111                         |
| 30 0006h | CONFIG4L | WRTAPP     | —           | —           | SAFEN        | BBEN    | BBSIZE<2:0>  |           |          | 1111 1111                         |
| 30 0007h | CONFIG4H | —          | —           | LVP         | —            | WRTSAF  | WRTD         | WRTC      | WRTB     | 1111 1111                         |
| 30 0008h | CONFIG5L | —          | —           | —           | —            | —       | —            | —         | CP       | 1111 1111                         |
| 30 0009h | CONFIG5H | —          | —           | —           | —            | —       | —            | —         | —        | 1111 1111                         |



## 5.3 Code Protection

Code protection allows the device to be protected from external access. Program memory protection and data memory are controlled through the  $\overline{\text{CP}}$  Configuration bit. Internal access to the program memory is unaffected by code protection setting.

The entire program memory space and Data EEPROM is protected from external reads and writes by the CP bit in Configuration Words. When CP = 0, external reads and writes of memory are inhibited and a read will return all '0's. The CPU can continue to read program memory and data EEPROM, regardless of the protection bit settings. Self-writing the program memory or Data EEPROM is dependent upon the write protection settings.

## 5.4 User ID

Eight words in the memory space (200000h-20000Fh) are designated as ID locations where the user can store checksum or other code identification numbers. These locations are readable and writable during normal execution. See [Section 13.2 “Device Information Area, Device Configuration Area, User ID, Device ID and Configuration Word Access”](#) for more information on accessing these memory locations. For more information on checksum calculation, see the “*PIC18(L)F25/26K83 Memory Programming Specification*” (DS40001927).

## 5.5 Device ID and Revision ID

The 16-bit device ID word is located at 3F FFEh and the 16-bit revision ID is located at 3F FFFCh. These locations are read-only and cannot be erased or modified.

Development tools, such as device programmers and debuggers, may be used to read the Device ID, Revision ID and Configuration Words. Refer to [13.0 “Nonvolatile Memory \(NVM\) Control”](#) for more information on accessing these locations.

## 5.6 Register Definitions: Device ID and Revision ID

### REGISTER 5-11: DEVICE ID: DEVICE ID REGISTER

|           |   |   |   |       |   |   |   |
|-----------|---|---|---|-------|---|---|---|
| R         | R | R | R | R     | R | R | R |
| DEV<15:8> |   |   |   |       |   |   |   |
| bit 15    |   |   |   | bit 8 |   |   |   |

|          |   |   |   |       |   |   |   |
|----------|---|---|---|-------|---|---|---|
| R        | R | R | R | R     | R | R | R |
| DEV<7:0> |   |   |   |       |   |   |   |
| bit 7    |   |   |   | bit 0 |   |   |   |

#### Legend:

R = Readable bit      '1' = Bit is set      0' = Bit is cleared      x = Bit is unknown

bit 15-0

**DEV<15:0>:** Device ID bits

| Device       | Device ID |
|--------------|-----------|
| PIC18F25K83  | 6EE0h     |
| PIC18F26K83  | 6EC0h     |
| PIC18LF25K83 | 6F20h     |
| PIC18LF26K83 | 6F00h     |

## REGISTER 5-12: REVISION ID: REVISION ID REGISTER

|        |   |   |   |             |   |   |   |
|--------|---|---|---|-------------|---|---|---|
| R      | R | R | R | R           | R | R | R |
| 1      | 0 | 1 | 0 | MJRREV<5:2> |   |   |   |
| bit 15 |   |   |   | bit 8       |   |   |   |

|             |   |             |   |   |   |   |   |
|-------------|---|-------------|---|---|---|---|---|
| R           | R | R           | R | R | R | R | R |
| MJRREV<1:0> |   | MNRREV<5:0> |   |   |   |   |   |
| bit 7       |   | bit 0       |   |   |   |   |   |

### Legend:

R = Readable bit      '1' = Bit is set      0' = Bit is cleared      x = Bit is unknown

bit 15-12      **Read as '1010'**

These bits are fixed with value '1010' for all devices in this family.

bit 11-6      **MJRREV<5:0>**: Major Revision ID bits

These bits are used to identify a major revision. A major revision is indicated by revision (A0, B0, C0, etc.)

Revision A = 0b00 0000

bit 5-0      **MNRREV<5:0>**: Minor Revision ID bits

These bits are used to identify a minor revision.

Revision A0 = 0b00 0000

## 5.7 Device Information Area

The Device Information Area (DIA) is a dedicated region in the program memory space. The DIA contains the calibration data for the internal temperature indicator module, stores the Microchip Unique Identifier words and the Fixed Voltage Reference voltage readings measured in mV.

The complete DIA table is shown in [Table 5-3: Device Information Area](#), followed by a description of each region and its functionality. The data is mapped from 3F0000h to 3F003Fh in the PIC18(L)F25/26K83 family. These locations are read-only and cannot be erased or modified by the user. The data is programmed into the device during manufacturing.

**TABLE 5-3: DEVICE INFORMATION AREA**

| Address Range   | Name of Region        | Standard Device Information                                      |
|-----------------|-----------------------|------------------------------------------------------------------|
| 3F0000h-3F000Bh | MUI0                  | Microchip Unique Identifier (6 Words)                            |
|                 | MUI1                  |                                                                  |
|                 | MUI2                  |                                                                  |
|                 | MUI3                  |                                                                  |
|                 | MUI4                  |                                                                  |
|                 | MUI5                  |                                                                  |
| 3F000Ch-3F000Fh | MUI6                  | Unassigned (2 Words)                                             |
|                 | MUI7                  |                                                                  |
| 3F0010h-3F0023h | EUI0                  | Optional External Unique Identifier (10 Words)                   |
|                 | EUI1                  |                                                                  |
|                 | EUI2                  |                                                                  |
|                 | EUI3                  |                                                                  |
|                 | EUI4                  |                                                                  |
|                 | EUI5                  |                                                                  |
|                 | EUI6                  |                                                                  |
|                 | EUI7                  |                                                                  |
|                 | EUI8                  |                                                                  |
|                 | EUI9                  |                                                                  |
| 3F0024h-3F0025h | TSLR1                 | Unassigned (1 Word)                                              |
| 3F0026h-3F0027h | TSLR2                 | Temperature Indicator ADC reading at @ 90°C (low range setting)  |
| 3F0028h-3F0029h | TSLR3                 | Unassigned (1 word)                                              |
| 3F002Ah-3F002Bh | TSHR1                 | Unassigned (1 Word)                                              |
| 3F002Ch-3F002Dh | TSHR2                 | Temperature Indicator ADC reading at @ 90°C (high range setting) |
| 3F002Eh-3F002Fh | TSHR3                 | Unassigned (1 Word)                                              |
| 3F0030h-3F0031h | FVRA1X                | ADC FVR1 Output voltage for 1x setting (in mV)                   |
| 3F0032h-3F0033h | FVRA2X                | ADC FVR1 Output Voltage for 2x setting (in mV)                   |
| 3F0034h-3F0035h | FVRA4X <sup>(1)</sup> | ADC FVR1 Output Voltage for 4x setting (in mV)                   |
| 3F0036h-3F0037h | FVRC1X                | Comparator FVR2 output voltage for 1x setting (in mV)            |
| 3F0038h-3F0039h | FVRC2X                | Comparator FVR2 output voltage for 2x setting (in mV)            |
| 3F003Ah-3F003Bh | FVRC4X                | Comparator FVR2 output voltage for 4x setting (in mV)            |
| 3F003Ch-3F003Fh |                       | Unassigned (2 Words)                                             |

**Note 1:** Value not present on LF devices.

## 5.7.1 MICROCHIP UNIQUE IDENTIFIER (MUI)

The PIC18(L)F25/26K83 devices are individually encoded during final manufacturing with a Microchip Unique Identifier, or MUI. The MUI cannot be user-erased. This feature allows for manufacturing traceability of Microchip Technology devices in applications where this is a required. It may also be used by the application manufacturer for a number of functions that require unverified unique identification, such as:

- Tracking the device
- Unique serial number

The MUI consists of six program words. When read together, these fields form a unique identifier. The MUI is stored in nine read-only locations, located between 3F0000h to 3F000Fh in the DIA space. [Table 5-3](#) lists the addresses of the identifier words.

**Note:** For applications that require verified unique identification, contact your Microchip Technology sales office to create a Serialized Quick Turn Programming<sup>SM</sup> option.

## 5.7.2 EXTERNAL UNIQUE IDENTIFIER (EUI)

The EUI data is stored at locations 3F0010h to 3F0023h in the Program Memory region. This region is an optional space for placing application specific information. The data is coded per customer requirements during manufacturing.

**Note:** Data is stored in this address range on receiving a request from the customer. The customer may contact the local sales representative, or Field Applications Engineer, and provide them the unique identifier information that is supposed to be stored in this region.

## 5.7.3 ANALOG-TO-DIGITAL CONVERSION DATA OF THE TEMPERATURE SENSOR

The purpose of the Temperature Sensor module is to provide a temperature-dependent voltage that can be measured by an analog module, see [Section 36.0 “Temperature Indicator Module”](#).

The DIA table contains the internal ADC measurement values of the Temperature sensor for Low and High range at fixed points of reference. The values are measured during test and are unique to each device. The measurement data is stored in the DIA memory region as hexadecimal numbers corresponding to the ADC conversion result. The calibration data can be used to plot the approximate sensor output voltage, VTSense vs. Temperature curve without having to make calibration measurements in the application. For more information on the operation of the Temperature Sensor, refer to [Section 36.0 “Temperature Indicator Module”](#).

- **TSLR2:** Address 3F0026h to 3F0027h store the measurements for the low-range setting of the Temperature Sensor at  $V_{DD} = 3V$ .
- **TSHR2:** Address 3F002Ch to 3F002Dh store the measurements for the High Range setting of the Temperature Sensor at  $V_{DD} = 3V$ .
- The stored measurements are made by the device ADC using the internal  $V_{REF} = 2.048V$ .

## 5.7.4 FIXED VOLTAGE REFERENCE DATA

The DIA stores measured FVR voltages for this device in mV for the different buffer settings of 1x, 2x or 4x at Program Memory locations 3F0030h to 3F003Bh. For more information on the FVR, refer to [Section 35.0 “Fixed Voltage Reference \(FVR\)”](#).

- FVRA1X stores the value of ADC FVR1 Output voltage for 1x setting (in mV)
- FVRA2X stores the value of ADC FVR1 Output Voltage for 2x setting (in mV)
- FVRA4X stores the value of ADC FVR1 Output Voltage for 4x setting (in mV)
- FVRC1X stores the value of Comparator FVR2 output voltage for 2x setting (in mV)
- FVRC2X stores the value of Comparator FVR2 output voltage for 2x setting (in mV)
- FVRC4X stores the value of Comparator FVR2 output voltage for 4x setting (in mV)

## 5.8 Device Configuration Information

The Device Configuration Information (DCI) is a dedicated region in the program memory space mapped from 3FFF00h to 3FFF09h. The data stored in these locations is read-only and cannot be erased.

Refer to [Table 5-4: Device Configuration Information for PIC18\(L\)F25/26K83](#) for the complete DCI table address and description. The DCI holds information about the device which is useful for programming and bootloader applications.

The erase size is the minimum erasable unit in the PFM, expressed as rows. The total device Flash memory capacity is (Row Size \* Number of rows)

**TABLE 5-4: DEVICE CONFIGURATION INFORMATION FOR PIC18(L)F25/26K83**

| ADDRESS           | Name  | DESCRIPTION                     | VALUE          |                | UNITS |
|-------------------|-------|---------------------------------|----------------|----------------|-------|
|                   |       |                                 | PIC18(L)F25K83 | PIC18(L)F26K83 |       |
| 3F FF00h-3F FF01h | ERSIZ | Erase Row Size                  | 64             | 64             | Words |
| 3F FF02h-3F FF03h | WLSIZ | Number of write latches per row | 128            | 128            | Bytes |
| 3F FF04h-3F FF05h | URSIZ | Number of User Rows             | 256            | 512            | Rows  |
| 3F FF06h-3F FF07h | EESIZ | Data EEPROM memory size         | 1024           | 1024           | Bytes |
| 3F FF08h-3F FF09h | PCNT  | Pin Count                       | 28             | 28             | Pins  |

## 6.0 RESETS

There are multiple ways to reset this device:

- Power-on Reset (POR)
- Brown-out Reset (BOR)
- Low-Power Brown-Out Reset (LPBOR)
- MCLR Reset
- WDT Reset
- RESET instruction
- Stack Overflow
- Stack Underflow
- Programming mode exit
- Memory Execution Violation Reset ( $\overline{\text{MEMV}}$ )

To allow VDD to stabilize, an optional Power-up Timer can be enabled to extend the Reset time after a BOR or POR event.

A simplified block diagram of the On-Chip Reset Circuit is shown in [Figure 6-1](#).

**FIGURE 6-1: SIMPLIFIED BLOCK DIAGRAM OF ON-CHIP RESET CIRCUIT**

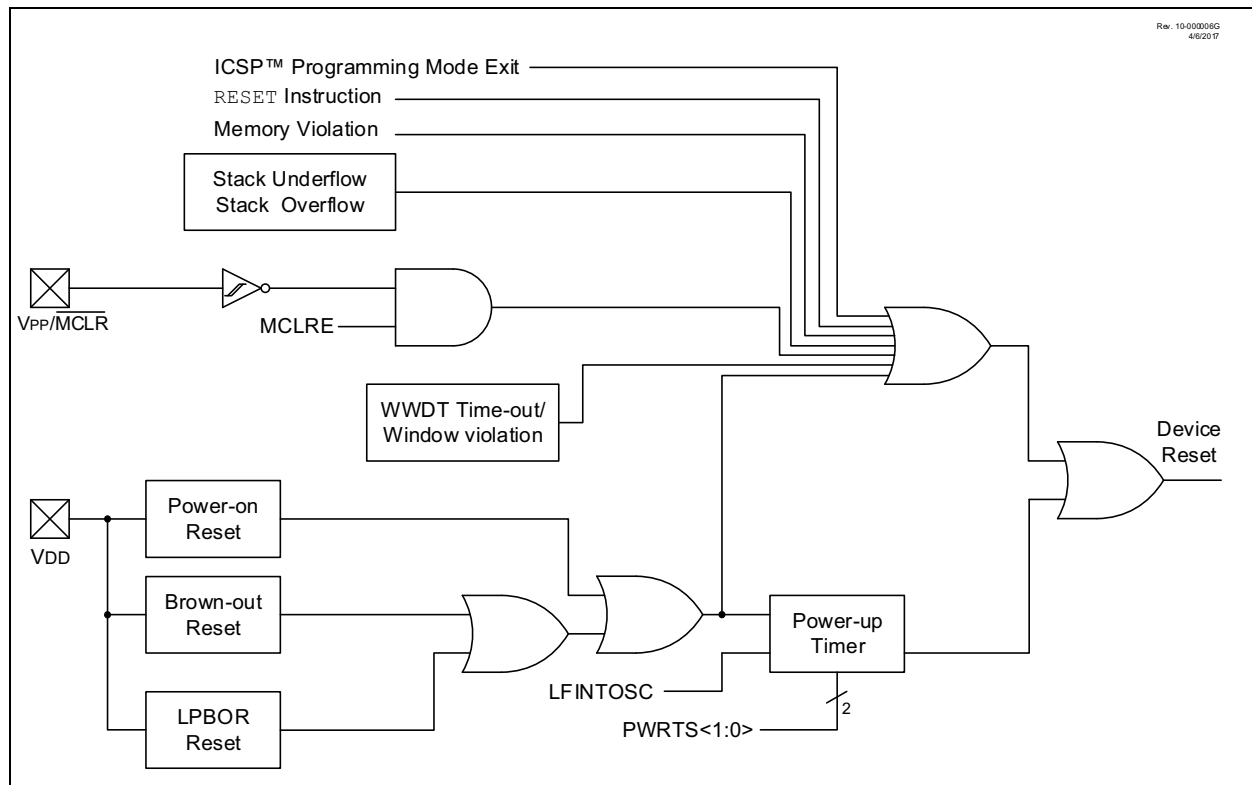
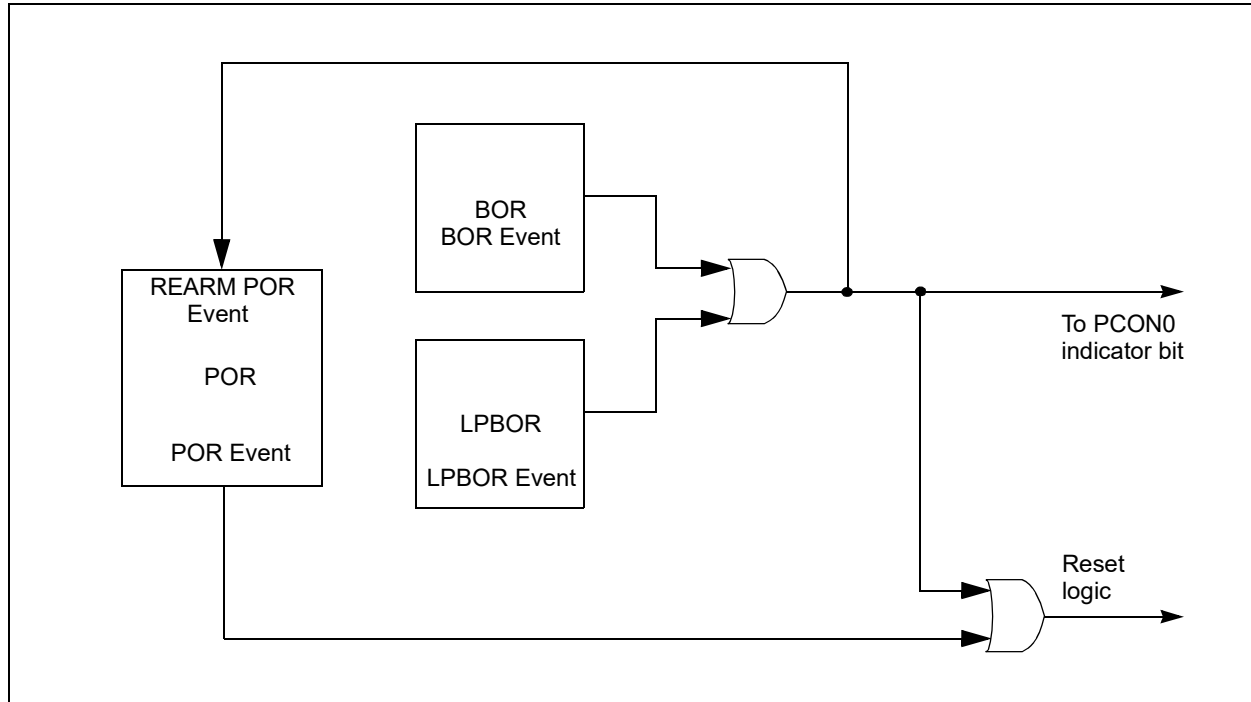


**FIGURE 6-2: LPBOR, BOR, POR RELATIONSHIP**





## 6.1 Power-on Reset (POR)

The POR circuit holds the device in Reset until VDD has reached an acceptable level for minimum operation. Slow rising VDD, fast operating speeds or analog performance may require greater than minimum VDD. The PWRT, BOR or MCLR features can be used to extend the start-up period until all device operation conditions have been met.

## 6.2 Brown-out Reset (BOR)

The BOR circuit holds the device in Reset when VDD reaches a selectable minimum level. Between the POR and BOR, complete voltage range coverage for execution protection can be implemented.

The Brown-out Reset module has four operating modes controlled by the BOREN<1:0> bits in Configuration Words. The four operating modes are:

- BOR is always on
- BOR is off when in Sleep
- BOR is controlled by software
- BOR is always off

Refer to [Table 6-1](#) for more information.

The Brown-out Reset voltage level is selectable by configuring the BORV<1:0> bits in Configuration Words.

A VDD noise rejection filter prevents the BOR from triggering on small events. If VDD falls below VBOR for a duration greater than parameter TBORDC, the device will reset. See [Table 45-11](#) for more information.

### 6.2.1 BOR IS ALWAYS ON

When the BOREN bits of Configuration Words are programmed to '11', the BOR is always on. The device start-up will be delayed until the BOR is ready and VDD is higher than the BOR threshold.

BOR protection is active during Sleep. The BOR does not delay wake-up from Sleep.

### 6.2.2 BOR IS OFF IN SLEEP

When the BOREN bits of Configuration Words are programmed to '10', the BOR is on, except in Sleep. The device start-up will be delayed until the BOR is ready and VDD is higher than the BOR threshold.

BOR protection is not active during Sleep. The device wake-up will be delayed until the BOR is ready.

### 6.2.3 BOR CONTROLLED BY SOFTWARE

When the BOREN bits of Configuration Words are programmed to '01', the BOR is controlled by the SBOREN bit of the BORCON register. The device start-up is not delayed by the BOR ready condition or the VDD level.

BOR protection begins as soon as the BOR circuit is ready. The status of the BOR circuit is reflected in the BORRDY bit of the BORCON register.

BOR protection is unchanged by Sleep.

### 6.2.4 BOR AND BULK ERASE

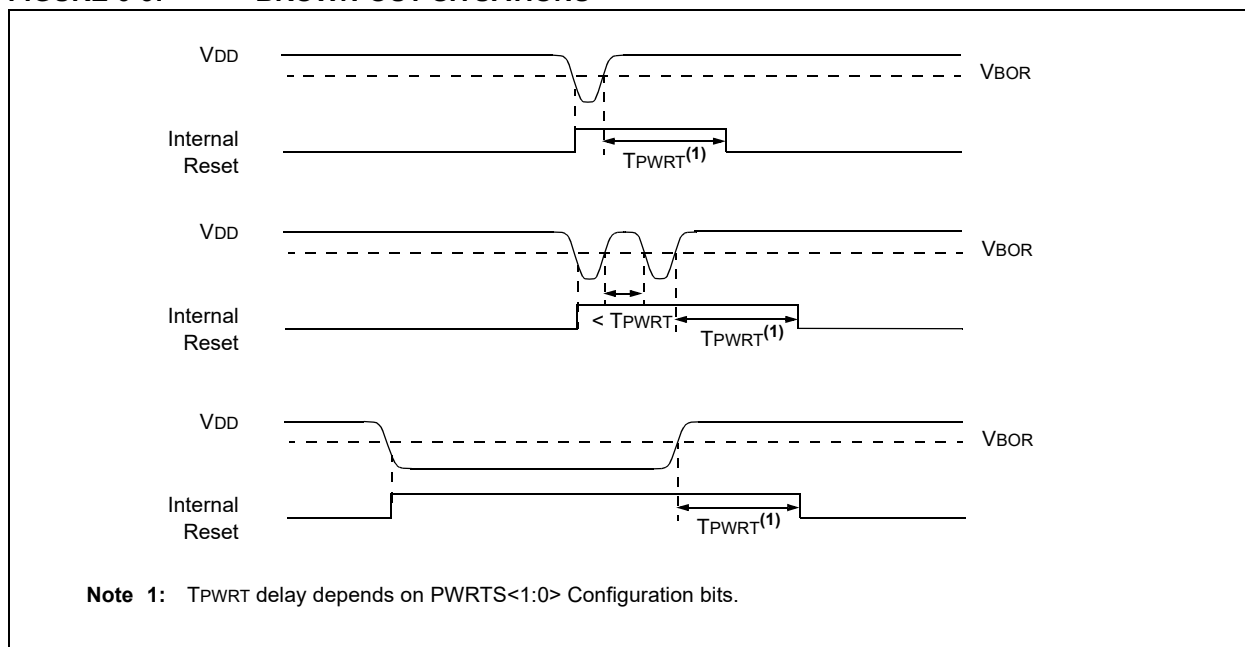
BOR is forced ON during PFM Bulk Erase operations to make sure that a safe erase voltage is maintained for a successful erase cycle.

During Bulk Erase, the BOR is enabled at 2.45V for F and LF devices, even if it is configured to some other value. If VDD falls, the erase cycle will be aborted, but the device will not be reset.

**TABLE 6-1: BOR OPERATING MODES**

| BOREN<1:0> | SBOREN | Device Mode | BOR Mode  | Instruction Execution upon:          |                                      |
|------------|--------|-------------|-----------|--------------------------------------|--------------------------------------|
|            |        |             |           | Release of POR                       | Wake-up from Sleep                   |
| 11         | X      | X           | Active    | Wait for release of BOR (BORRDY = 1) | Begins immediately                   |
| 10         | X      | Awake       | Active    | Wait for release of BOR (BORRDY = 1) | N/A                                  |
|            |        | Sleep       | Hibernate | N/A                                  | Wait for release of BOR (BORRDY = 1) |
| 01         | 1      | X           | Active    | Wait for release of BOR (BORRDY = 1) | Begins immediately                   |
|            | 0      | X           | Hibernate |                                      |                                      |
| 00         | X      | X           | Disabled  | Begins immediately                   |                                      |

**FIGURE 6-3: BROWN-OUT SITUATIONS**



## 6.3 Register Definitions: BOR Control

**REGISTER 6-1: BORCON: BROWN-OUT RESET CONTROL REGISTER**

| R/W-1/u | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R-q/u  |
|---------|-----|-----|-----|-----|-----|-----|--------|
| SBOREN  | —   | —   | —   | —   | —   | —   | BORRDY |
| bit 7   |     |     |     |     |     |     | bit 0  |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

q = Value depends on condition

bit 7 **SBOREN:** Software Brown-out Reset Enable bit

If **BOREN** ≠ 01:

SBOREN is read/write, but has no effect on the BOR.

If **BOREN** = 01:

1 = BOR Enabled

0 = BOR Disabled

bit 6-1 **Unimplemented:** Read as '0'

bit 0 **BORRDY:** Brown-out Reset Circuit Ready Status bit

1 = The Brown-out Reset Circuit is active and armed

0 = The Brown-out Reset Circuit is disabled or is warming up

## 6.4 Low-Power Brown-out Reset (LPBOR)

The Low-Power Brown-out Reset (LPBOR) provides an additional BOR circuit for low power operation. Refer to [Figure 6-2](#) to see how the BOR interacts with other modules.

The LPBOR is used to monitor the external VDD pin. When too low of a voltage is detected, the device is held in Reset.

### 6.4.1 ENABLING LPBOR

The LPBOR is controlled by the  $\overline{\text{LPBOREN}}$  bit of Configuration Word 2L. When the device is erased, the LPBOR module defaults to disabled.

#### 6.4.1.1 LPBOR Module Output

The output of the LPBOR module is a signal indicating whether or not a Reset is to be asserted. This signal is OR'd together with the Reset signal of the BOR module to provide the generic  $\overline{\text{BOR}}$  signal, which goes to the PCON0 register and to the power control block.

## 6.5 $\overline{\text{MCLR}}$

The  $\overline{\text{MCLR}}$  is an optional external input that can reset the device. The  $\overline{\text{MCLR}}$  function is controlled by the MCLRE bit of Configuration Words and the LVP bit of Configuration Words ([Table 6-2](#)). The  $\overline{\text{RMCLR}}$  bit in the PCON0 register will be set to '0' if a MCLR Reset has occurred.

**TABLE 6-2:  $\overline{\text{MCLR}}$  CONFIGURATION**

| MCLRE | LVP | $\overline{\text{MCLR}}$ |
|-------|-----|--------------------------|
| x     | 1   | Enabled                  |
| 1     | 0   | Enabled                  |
| 0     | 0   | Disabled                 |

### 6.5.1 $\overline{\text{MCLR}}$ ENABLED

When  $\overline{\text{MCLR}}$  is enabled and the pin is held low, the device is held in Reset. The  $\overline{\text{MCLR}}$  pin is connected to VDD through an internal weak pull-up.

The device has a noise filter in the  $\overline{\text{MCLR}}$  Reset path. The filter will detect and ignore small pulses.

**Note:** An internal Reset event ( $\overline{\text{RESET}}$  instruction, BOR, WWDT, POR stack), does not drive the  $\overline{\text{MCLR}}$  pin low.

### 6.5.2 $\overline{\text{MCLR}}$ DISABLED

When  $\overline{\text{MCLR}}$  is disabled, the  $\overline{\text{MCLR}}$  pin becomes input-only and pin functions such as internal weak pull-ups are under software control. See [Section 16.2 "I/O Priorities"](#) for more information.

## 6.6 Windowed Watchdog Timer (WWDT) Reset

The Windowed Watchdog Timer generates a Reset if the firmware does not issue a  $\overline{\text{CLRWDT}}$  instruction within the time-out period or window set. The  $\overline{\text{TO}}$  and  $\overline{\text{PD}}$  bits in the STATUS register and the  $\overline{\text{RWDT}}$  bit in the PCON0 register are changed to indicate a WWDT Reset. The WDTWV bit in the PCON0 register indicates if the WDT Reset has occurred due to a time out or a window violation. See [Section 11.0 "Windowed Watchdog Timer \(WWDT\)"](#) for more information.

## 6.7 $\overline{\text{RESET}}$ Instruction

A  $\overline{\text{RESET}}$  instruction will cause a device Reset. The  $\overline{\text{RI}}$  bit in the PCON0 register will be set to '0'. See [Table 6-3](#) for default conditions after a  $\overline{\text{RESET}}$  instruction has occurred.

## 6.8 Stack Overflow/Underflow Reset

The device can reset when the Stack Overflows or Underflows. The STKOVF or STKUNF bits of the PCON0 register indicate the Reset condition. These Resets are enabled by setting the STVREN bit in Configuration Words. See [Section 4.2.5 "Return Address Stack"](#) for more information.

## 6.9 Programming Mode Exit

Upon exit of Programming mode, the device will behave as if a POR has just occurred.

## 6.10 Power-up Timer (PWRT)

The Power-up Timer provides a selected time-out duration on POR or Brown-out Reset.

The device is held in Reset as long as PWRT is active. The PWRT delay allows additional time for the VDD to rise to an acceptable level. The Power-up Timer is selected by setting the PWRTS<1:0> Configuration bits, appropriately.

The Power-up Timer starts after the release of the POR and BOR/LPBOR if enabled, as shown in [Figure 6-1](#).

## 6.11 Start-up Sequence

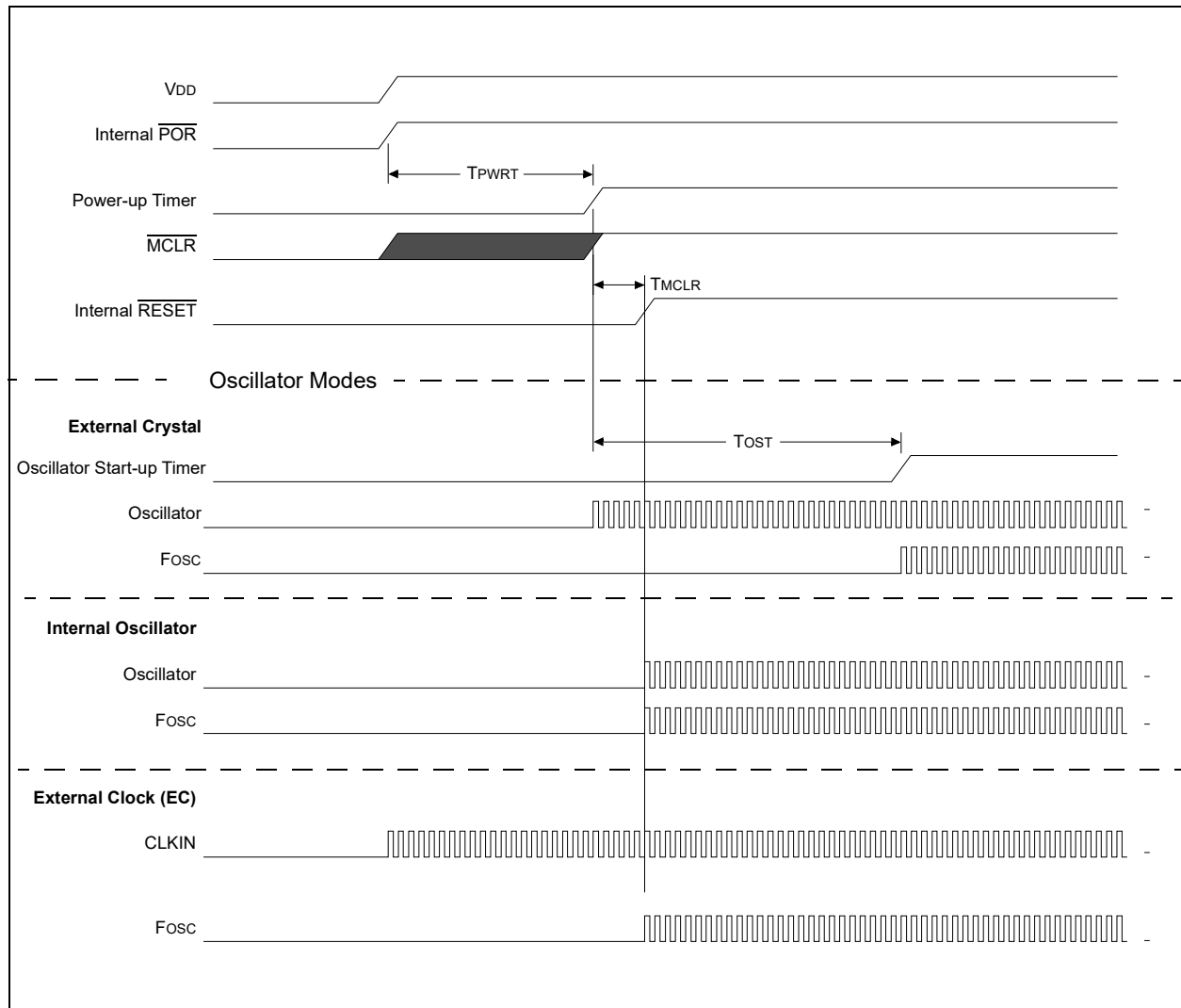
Upon the release of a POR or BOR, the following must occur before the device will begin executing:

1. Power-up Timer runs to completion (if enabled).
2. Oscillator start-up timer runs to completion (if required for selected oscillator source).
3.  $\overline{\text{MCLR}}$  must be released (if enabled).

The total time out will vary based on oscillator configuration and Power-up Timer configuration. See [Section 7.0 “Oscillator Module \(with Fail-Safe Clock Monitor\)”](#) for more information.

The Power-up Timer and oscillator start-up timer run independently of  $\overline{\text{MCLR}}$  Reset. If  $\overline{\text{MCLR}}$  is kept low long enough, the Power-up Timer and oscillator Start-up Timer will expire. Upon bringing  $\overline{\text{MCLR}}$  high, the device will begin execution after 10  $F_{\text{OSC}}$  cycles (see [Figure 6-4](#)). This is useful for testing purposes or to synchronize more than one device operating in parallel.

**FIGURE 6-4: RESET START-UP SEQUENCE**



## 6.11.1 MEMORY EXECUTION VIOLATION

If the CPU executes outside the valid execution area, a memory execution violation Reset occurs.

The invalid execution areas are:

1. Addresses outside implemented program memory (see [Table 5-1](#)).
2. Storage Area Flash (SAF) inside program memory, if it is enabled.

When a memory execution violation is generated, flag MEMV is cleared in PCON1 ([Register 6-3](#)) to signal the cause of Reset. It needs to be set in the user code after a memory execution violation Reset has occurred to detect further violation Resets.

## 6.12 Determining the Cause of a Reset

Upon any Reset, multiple bits in the STATUS and PCON0 registers are updated to indicate the cause of the Reset. [Table 6-3](#) shows the Reset conditions of these registers.

**TABLE 6-3: RESET CONDITION FOR SPECIAL REGISTERS**

| Condition                          | Program Counter | STATUS Register <sup>(1,2)</sup> | PCON0 Register | PCON1 Register |
|------------------------------------|-----------------|----------------------------------|----------------|----------------|
| Power-on Reset                     | 0               | -110 0000                        | 0011 110x      | ---- --1-      |
| Brown-out Reset                    | 0               | -110 0000                        | 0011 11u0      | ---- --1-      |
| MCLR Reset during normal operation | 0               | -uuu uuuu                        | uuuu 0uuu      | ---- --u-      |
| MCLR Reset during Sleep            | 0               | -10u uuuu                        | uuuu 0uuu      | ---- --u-      |
| WWDT Time-out Reset                | 0               | -0uu uuuu                        | uuu0 uuuu      | ---- --u-      |
| WWDT Window Violation Reset        | 0               | -uuu uuuu                        | uu0u uuuu      | ---- --u-      |
| RESET Instruction Executed         | 0               | -uuu uuuu                        | uuuu u0uu      | ---- --u-      |
| Stack Overflow Reset (STVREN = 1)  | 0               | -uuu uuuu                        | 1uuu uuuu      | ---- --u-      |
| Stack Underflow Reset (STVREN = 1) | 0               | -uuu uuuu                        | u1uu uuuu      | ---- --u-      |
| Memory Violation Reset             | 0               | -uuu uuuu                        | uuuu uuuu      | ---- --0-      |

**Legend:** u = unchanged, x = unknown, – = unimplemented bit, reads as ‘0’.

**Note 1:** If a Status bit is not implemented, that bit will be read as ‘0’.

**2:** Status bits Z, C, DC are reset by POR/BOR, but not defined by the Resets module ([Register 4-2](#)).

## 6.13 Power Control (PCON0/PCON1) Register

The Power Control (PCON0/PCON1) register contains flag bits to differentiate between a:

- Brown-out Reset ( $\overline{\text{BOR}}$ )
- Power-on Reset ( $\overline{\text{POR}}$ )
- Reset Instruction Reset ( $\overline{\text{RI}}$ )
- MCLR Reset ( $\overline{\text{RMCLR}}$ )
- Watchdog Timer Reset ( $\overline{\text{RWDI}}$ )
- Watchdog Window Violation ( $\overline{\text{WDTWV}}$ )
- Stack Underflow Reset (STKUNF)
- Stack Overflow Reset (STKOVF)
- Memory Violation Reset ( $\overline{\text{MEMV}}$ )

The PCON0/1 register bits are shown in [Register 6-2](#) and [Register 6-3](#). Hardware will change the corresponding register bit during the Reset process; if the Reset was not caused by the condition, the bit remains unchanged ([Table 6-3](#)).

Software should reset the bit to the inactive state after restart (hardware will not reset the bit). Software may also set any PCON0 bit to the active state, so that user code may be tested, but no Reset action will be generated.

## 6.14 Register Definitions: Power Control

### REGISTER 6-2: PCON0: POWER CONTROL REGISTER 0

| R/W/HS-0/q | R/W/HS-0/q | R/W/HC-1/q | R/W/HC-1/q | R/W/HC-1/q | R/W/HC-1/q | R/W/HC-0/u | R/W/HC-q/u |
|------------|------------|------------|------------|------------|------------|------------|------------|
| STKOVF     | STKUNF     | WDTWV      | RWDT       | RMCLR      | RI         | POR        | BOR        |
| bit 7      |            |            |            |            |            |            | bit 0      |

#### Legend:

HC = Bit is cleared by hardware

HS = Bit is set by hardware

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-m/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

q = Value depends on condition

bit 7 **STKOVF**: Stack Overflow Flag bit

1 = A Stack Overflow occurred (more CALLs than fit on the stack)

0 = A Stack Overflow has not occurred or set to '0' by firmware

bit 6 **STKUNF**: Stack Underflow Flag bit

1 = A Stack Underflow occurred (more RETURNS than CALLs)

0 = A Stack Underflow has not occurred or set to '0' by firmware

bit 5 **WDTWV**: Watchdog Window Violation bit

1 = A WDT window violation has not occurred or set to '1' by firmware

0 = A CLRWD instruction was issued when the WDT Reset window was closed (set to '0' in hardware when a WDT window violation Reset occurs)

bit 4 **RWDT**: WDT Reset Flag bit

1 = A WDT overflow/time-out Reset has not occurred or set to '1' by firmware

0 = A WDT overflow/time-out Reset has occurred (set to '0' in hardware when a WDT Reset occurs)

bit 3 **RMCLR**: MCLR Reset Flag bit

1 = A MCLR Reset has not occurred or set to '1' by firmware

0 = A MCLR Reset has occurred (set to '0' in hardware when a MCLR Reset occurs)

bit 2 **RI**: RESET Instruction Flag bit

1 = A RESET instruction has not been executed or set to '1' by firmware

0 = A RESET instruction has been executed (set to '0' in hardware upon executing a RESET instruction)

bit 1 **POR**: Power-on Reset Status bit

1 = No Power-on Reset occurred or set to '1' by firmware

0 = A Power-on Reset occurred (set to '0' in hardware when a Power-on Reset occurs)

bit 0 **BOR**: Brown-out Reset Status bit

1 = No Brown-out Reset occurred or set to '1' by firmware

0 = A Brown-out Reset occurred (set to '0' in hardware when a Brown-out Reset occurs)



**REGISTER 6-3: PCON1: POWER CONTROL REGISTER 1**

|       |     |     |     |     |     |            |       |
|-------|-----|-----|-----|-----|-----|------------|-------|
| U-0   | U-0 | U-0 | U-0 | U-0 | U-0 | R/W/HC-1/u | U-0   |
| —     | —   | —   | —   | —   | —   | MEMV       | —     |
| bit 7 |     |     |     |     |     |            | bit 0 |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-m/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

q = Value depends on condition

bit 7-2 **Unimplemented:** Read as '0'

bit 1 **MEMV:** Memory Violation Flag bit

1 = No memory violation Reset occurred or set to '1' by firmware

0 = A memory violation Reset occurred (set to '0' in hardware when a memory violation occurs)

bit 0 **Unimplemented:** Read as '0'

**TABLE 6-4: SUMMARY OF REGISTERS ASSOCIATED WITH RESETS**

| Name   | Bit 7  | Bit 6  | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0  | Register on Page   |
|--------|--------|--------|-------|-------|-------|-------|-------|--------|--------------------|
| BORCON | SBOREN | —      | —     | —     | —     | —     | —     | BORRDY | <a href="#">75</a> |
| PCON0  | STKOVF | STKUNF | WDTWV | RWDT  | RMCLR | RI    | POR   | BOR    | <a href="#">80</a> |
| PCON1  | —      | —      | —     | —     | —     | —     | MEMV  | —      | <a href="#">81</a> |

**Legend:** — = unimplemented location, read as '0'. Shaded cells are not used by Resets.

## 7.0 OSCILLATOR MODULE (WITH FAIL-SAFE CLOCK MONITOR)

### 7.1 Overview

The oscillator module has multiple clock sources and selection features that allow it to be used in a wide range of applications while maximizing performance and minimizing power consumption. [Figure 7-1](#) illustrates a block diagram of the oscillator module.

Clock sources can be supplied from external oscillators, quartz-crystal resonators and ceramic resonators. In addition, the system clock source can be supplied from one of two internal oscillators and PLL circuits, with a choice of speeds selectable via software. Additional clock features include:

- Selectable system clock source between external or internal sources via software.
- Fail-Safe Clock Monitor (FSCM) designed to detect a failure of the external clock source (LP, XT, HS, ECH, ECM, ECL) and switch automatically to the internal oscillator.
- Oscillator Start-up Timer (OST) ensures stability of crystal oscillator sources.

The RSTOSC bits of Configuration Word 1 ([Register 5-1](#)) determine the type of oscillator that will be used when the device runs after Reset, including when it is first powered up.

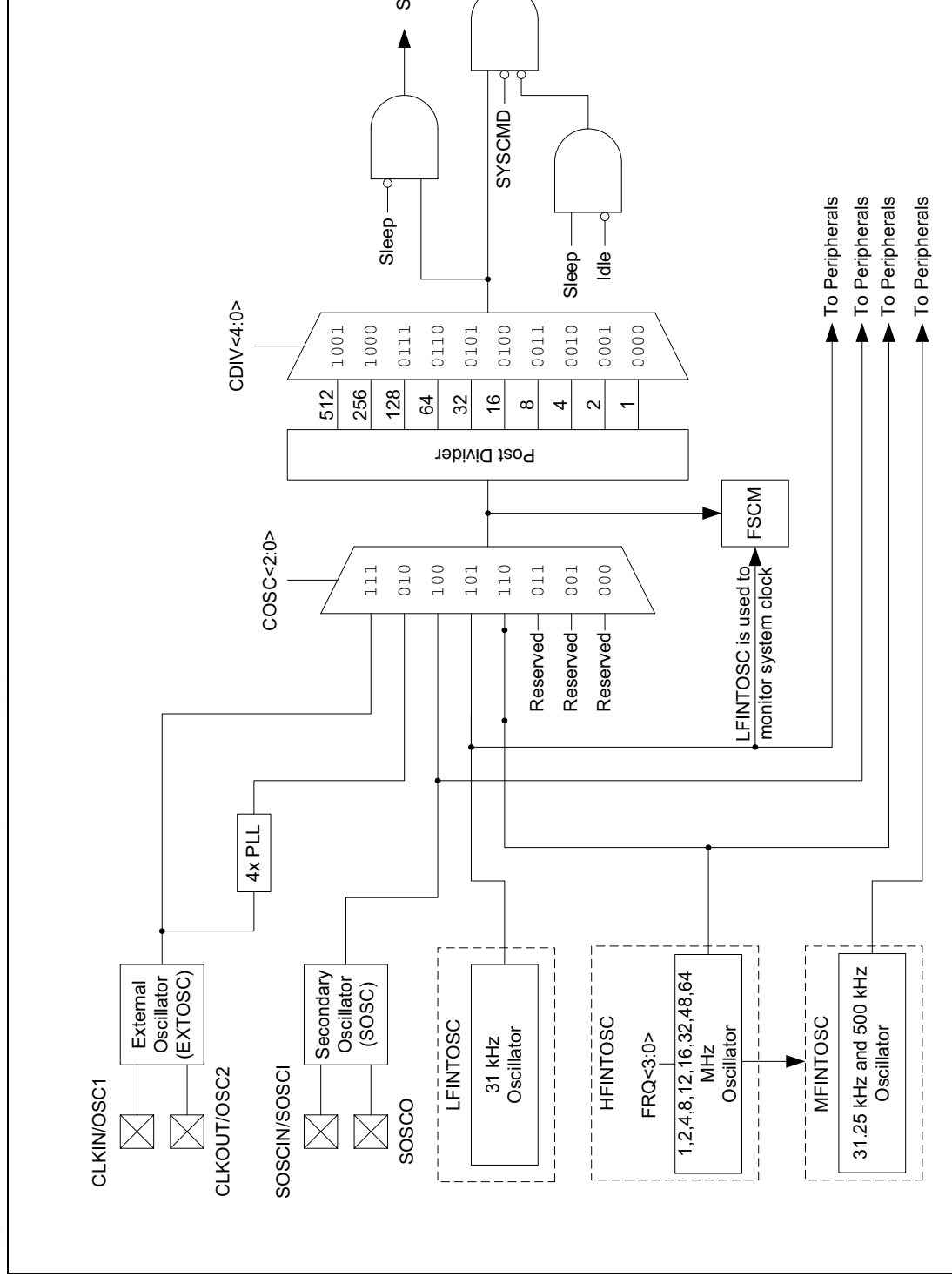
If an external clock source is selected, the FEXTOSC bits of Configuration Word 1 must be used in conjunction with the RSTOSC bits to select the External Clock mode.

The external oscillator module can be configured in one of the following clock modes, by setting the FEXTOSC<2:0> Configuration bits:

1. ECL – External Clock Low-Power mode
2. ECM – External Clock Medium Power mode
3. ECH – External Clock High-Power mode
4. LP – 32 kHz Low-Power Crystal mode.
5. XT – Medium Gain Crystal or Ceramic Resonator Oscillator mode
6. HS – High Gain Crystal or Ceramic Resonator mode

The ECH, ECM, and ECL Clock modes rely on an external logic level signal as the device clock source. The LP, XT, and HS Clock modes require an external crystal or resonator to be connected to the device. Each mode is optimized for a different frequency range. The internal oscillator block produces low and high-frequency clock sources, designated LFINTOSC and HFINTOSC. (see Internal Oscillator Block, [Figure 7-1](#)). Multiple device clock frequencies may be derived from these clock sources.

**FIGURE 7-1: SIMPLIFIED PIC® MCU CLOCK SOURCE BLOCK DIAGRAM**



## 7.2 Clock Source Types

Clock sources can be classified as external or internal.

External clock sources rely on external circuitry for the clock source to function. Examples are: oscillator modules (ECH, ECM, ECL mode), quartz crystal resonators or ceramic resonators (LP, XT and HS modes).

Internal clock sources are contained within the oscillator module. The internal oscillator block has two internal oscillators that are used to generate internal system clock sources. The High-Frequency Internal Oscillator (HFINTOSC) can produce 1, 2, 4, 8, 12, 16, 32, 48 and 64 MHz clock. The frequency can be controlled through the OSCFRQ register (Register 7-5). The Low-Frequency Internal Oscillator (LFINTOSC) generates a fixed 31 kHz frequency.

A 4x PLL is provided that can be used with an external clock. When used with the EXTOSC the 4x PLL has input frequency limitations. See Section 7.2.1.4 “4x PLL” for more details.

The system clock can be selected between external or internal clock sources via the NOSC bits in the OSCCON1 register. See Section 7.3 “Clock Switching” for additional information. The system clock can be made available on the OSC2/CLKOUT pin for any of the modes that do not use the OSC2 pin. The clock out functionality is governed by the CLKOUTEN bit in the CONFIG1H register (Register 5-2). If enabled, the clock out signal is always at a frequency of Fosc/4.

### 7.2.1 EXTERNAL CLOCK SOURCES

An external clock source can be used as the device system clock by performing one of the following actions:

- Program the RSTOSC<2:0> and FEXTOSC<2:0> bits in the Configuration Words to select an external clock source that will be used as the default system clock upon a device Reset.
- Write the NOSC<2:0> and NDIV<3:0> bits in the OSCCON1 register to switch the system clock source.

See Section 7.3 “Clock Switching” for more information.

#### 7.2.1.1 EC Mode

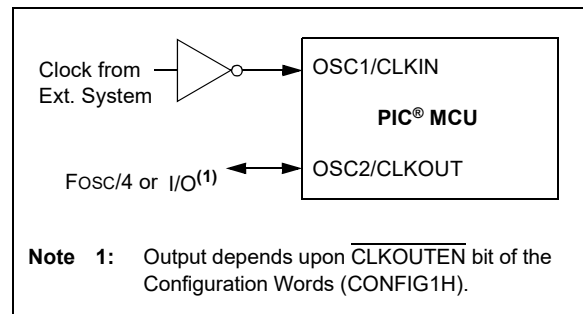
The External Clock (EC) mode allows an externally generated logic level signal to be the system clock source. When operating in this mode, an external clock source is connected to the OSC1 input. OSC2/CLKOUT is available for general purpose I/O or CLKOUT. Figure 7-2 shows the pin connections for EC mode.

EC mode has three power modes to select from through Configuration Words:

- ECH – High power
- ECM – Medium power
- ECL – Low power

The Oscillator Start-up Timer (OST) is disabled when EC mode is selected. Therefore, there is no delay in operation after a Power-on Reset (POR) or wake-up from Sleep. Because the PIC® MCU design is fully static, stopping the external clock input will have the effect of halting the device while leaving all data intact. Upon restarting the external clock, the device will resume operation as if no time had elapsed.

**FIGURE 7-2: EXTERNAL CLOCK (EC) MODE OPERATION**



#### 7.2.1.2 LP, XT, HS Modes

The LP, XT and HS modes support the use of quartz crystal resonators or ceramic resonators connected to OSC1 and OSC2 (Figure 7-3). The three modes select a low, medium or high gain setting of the internal inverter-amplifier to support various resonator types and speed.

**LP** Oscillator mode selects the lowest gain setting of the internal inverter-amplifier. LP mode current consumption is the least of the three modes. This mode is designed to drive only 32.768 kHz tuning-fork type crystals (watch crystals).

**XT** Oscillator mode selects the intermediate gain setting of the internal inverter-amplifier. XT mode current consumption is the medium of the three modes. This mode is best suited to drive resonators with a medium drive level specification (above 100 kHz - 8 MHz).

**HS** Oscillator mode selects the highest gain setting of the internal inverter-amplifier. HS mode current consumption is the highest of the three modes. This mode is best suited for resonators that require a high drive setting (above 8 MHz).

Figure 7-3 and Figure 7-4 show typical circuits for quartz crystal and ceramic resonators, respectively.

**FIGURE 7-3: QUARTZ CRYSTAL OPERATION (LP, XT OR HS MODE)**

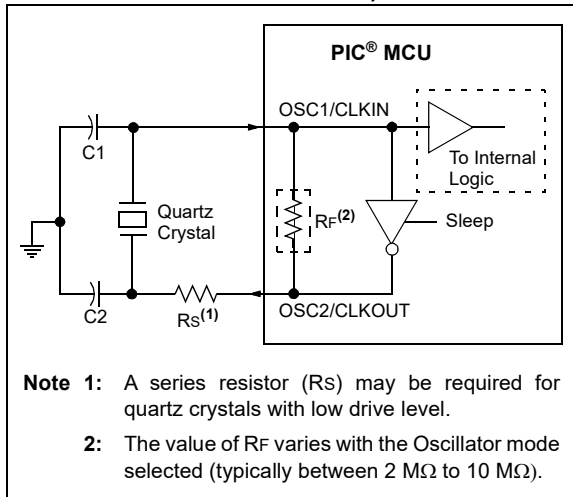
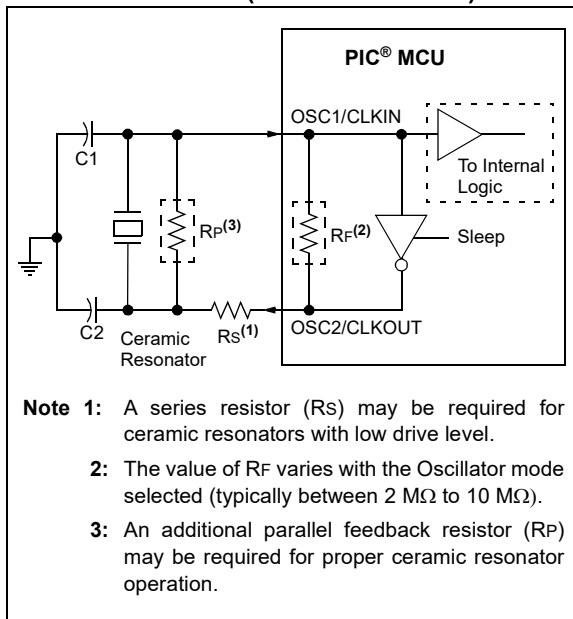


**FIGURE 7-4: CERAMIC RESONATOR OPERATION (XT OR HS MODE)**



## 7.2.1.3 Oscillator Start-up Timer (OST)

If the oscillator module is configured for LP, XT or HS modes, the Oscillator Start-up Timer (OST) counts 1024 oscillations from OSC1. This occurs following a Power-on Reset (POR), or a wake-up from Sleep. The OST ensures that the oscillator circuit, using a quartz crystal resonator or ceramic resonator, has started and is providing a stable system clock to the oscillator module.

## 7.2.1.4 4x PLL

The oscillator module contains a 4x PLL that can be used with the external clock sources to provide a system clock source. The input frequency for the PLL must fall within specifications. See the PLL Clock Timing Specifications in [Table 45-9](#).

The PLL can be enabled for use by one of two methods:

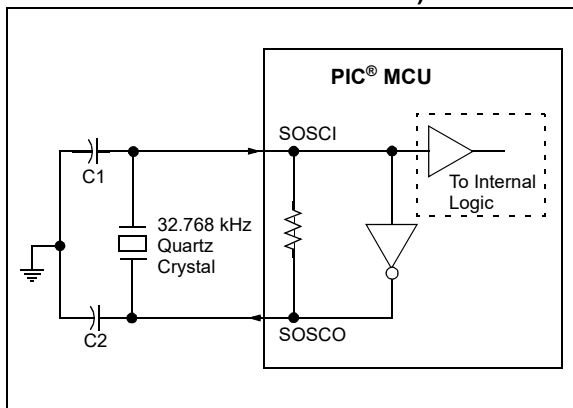
1. Program the RSTOSC bits in the Configuration Word 1 to 010 (enable EXTOSC with 4x PLL).
2. Write the NOSC bits in the OSCCON1 register to 010 (enable EXTOSC with 4x PLL).

## 7.2.1.5 Secondary Oscillator

The secondary oscillator is a separate oscillator block that can be used as an alternate system clock source. The secondary oscillator is optimized for 32.768 kHz, and can be used with an external crystal oscillator connected to the SOSC1 and SOSCO device pins, or an external clock source connected to the SOSCIN pin. The secondary oscillator can be selected during run-time using clock switching. Refer to [Section 7.3 “Clock Switching”](#) for more information.

Two power modes are available for the secondary oscillator. These modes are selected with the SOSCPWR (OSCCON3<6>). Clearing this bit selects the lower Crystal Gain mode which provides lowest microcontroller power consumption. Setting this bit enables a higher Gain mode to support faster crystal start-up or crystals with higher ESR.

**FIGURE 7-5: QUARTZ CRYSTAL OPERATION (SECONDARY OSCILLATOR)**



**Note 1:** Quartz crystal characteristics vary according to type, package and manufacturer. The user should consult the manufacturer data sheets for specifications and recommended application.

**2:** Always verify oscillator performance over the VDD and temperature range that is expected for the application.

**3:** For oscillator design assistance, reference the following Microchip Application Notes:

- AN826, “Crystal Oscillator Basics and Crystal Selection for PIC® and PIC® Devices” (DS00826)
- AN849, “Basic PIC® Oscillator Design” (DS00849)
- AN943, “Practical PIC® Oscillator Analysis and Design” (DS00943)
- AN949, “Making Your Oscillator Work” (DS00949)
- TB097, “Interfacing a Micro Crystal MS1V-T1K 32.768 kHz Tuning Fork Crystal to a PIC16F690/SS” (DS91097)
- AN1288, “Design Practices for Low-Power External Oscillators” (DS01288)

## 7.2.2 INTERNAL CLOCK SOURCES

The device may be configured to use the internal oscillator block as the system clock by performing one of the following actions:

- Program the RSTOSC<2:0> bits in Configuration Words to select the INTOSC clock source, which will be used as the default system clock upon a device Reset.
- Write the NOSC<2:0> bits in the OSCCON1 register to switch the system clock source to the internal oscillator during run-time. See [Section 7.3 “Clock Switching”](#) for more information.

In INTOSC mode, OSC1/CLKIN is available for general purpose I/O, provided that FEXTOSC is configured to ‘oscillator is not enabled’. OSC2/CLKOUT is available for general purpose I/O or CLKOUT.

The function of the OSC2/CLKOUT pin is determined by the CLKOUTEN bit in Configuration Words.

The internal oscillator block has two independent oscillators that can produce two internal system clock sources.

1. The **HFINTOSC** (High-Frequency Internal Oscillator) is factory-calibrated and operates from 1 to 64 MHz. The frequency of HFINTOSC can be selected through the OSCFRQ Frequency Selection register, and fine-tuning can be done via the OSCTUNE register.
2. The **LFINTOSC** (Low-Frequency Internal Oscillator) is factory-calibrated and operates at 31 kHz.

### 7.2.2.1 HFINTOSC

The High-Frequency Internal Oscillator (HFINTOSC) is a precision digitally-controlled internal clock source that produces a stable clock up to 64 MHz. The HFINTOSC can be enabled through one of the following methods:

- Programming the RSTOSC<2:0> bits in Configuration Word 1 to ‘110’ (FOSC = 1 MHz) or ‘000’ (FOSC = 64 MHz) to set the oscillator upon device Power-up or Reset.
- Write to the NOSC<2:0> bits of the OSCCON1 register during run-time. See [Section 7.3 “Clock Switching”](#) for more information.

The HFINTOSC frequency can be selected by setting the FRQ<3:0> bits of the OSCFRQ register.

The NDIV<3:0> bits of the OSCCON1 register allow for division of the HFINTOSC output from a range between 1:1 and 1:512.

### 7.2.2.2 MFINTOSC

The module provides two (500 kHz and 31.25 kHz) constant clock outputs. These clocks are digital divisors of the HFINTOSC clock. Dynamic divider logic is used to provide constant MFINTOSC clock rates for all settings of HFINTOSC.

The MFINTOSC cannot be used to drive the system but it is used to clock certain modules such as the Timers and WWDT.

## 7.2.2.3 Internal Oscillator Frequency Adjustment

The internal oscillator is factory-calibrated. This internal oscillator can be adjusted in software by writing to the OSCTUNE register ([Register 7-3](#)).

The default value of the OSCTUNE register is 00h. The value is a 6-bit two's complement number. A value of 1Fh will provide an adjustment to the maximum frequency. A value of 20h will provide an adjustment to the minimum frequency.

When the OSCTUNE register is modified, the oscillator frequency will begin shifting to the new frequency. Code execution continues during this shift. There is no indication that the shift has occurred.

OSCTUNE **does not affect** the LFINTOSC frequency. Operation of features that depend on the LFINTOSC clock source frequency, such as the Power-up Timer (PWRT), WWDT, Fail-Safe Clock Monitor (FSCM) and peripherals, are *not* affected by the change in frequency.

## 7.2.2.4 LFINTOSC

The Low-Frequency Internal Oscillator (LFINTOSC) is a factory-calibrated 31 kHz internal clock source.

The LFINTOSC is the frequency for the Power-up Timer (PWRT), Windowed Watchdog Timer (WWDT) and Fail-Safe Clock Monitor (FSCM).

The LFINTOSC is enabled through one of the following methods:

- Programming the RSTOSC<2:0> bits of Configuration Word 1 to enable LFINTOSC.
- Write to the NOSC<2:0> bits of the OSCCON1 register during run-time. See [Section 7.3, Clock Switching](#) for more information.

## 7.2.2.5 ADCRC

The ADCRC is an oscillator dedicated to the ADC<sup>2</sup> module. The ADCRC oscillator can be manually enabled using the ADOEN bit of the OSCEN register. The ADCRC runs at a fixed frequency of 600 kHz. ADCRC is automatically enabled if it is selected as the clock source for the ADC<sup>2</sup> module.



## 7.2.2.6 Oscillator Status and Manual Enable

The Ready status of each oscillator (including the ADCRC oscillator) is displayed in OSCSTAT (Register 7-4). The oscillators (but not the PLL) may be explicitly enabled through OSCEN (Register 7-7).

## 7.2.2.7 HFOR and MFOR Bits

The HFOR and MFOR bits indicate that the HFINTOSC and MFINTOSC is ready. These clocks are always valid for use at all times, but only accurate after they are ready.

When a new value is loaded into the OSCFRQ register, the HFOR and MFOR bits will clear, and set again when the oscillator is ready. During pending OSCFRQ changes the MFINTOSC clock will stall at a high or a low state, until the HFINTOSC resumes operation.

## 7.3 Clock Switching

The system clock source can be switched between external and internal clock sources via software using the New Oscillator Source (NOSC) bits of the OSCCON1 register. The following clock sources can be selected using the following:

- External oscillator
- Internal Oscillator Block (INTOSC)

**Note:** The Clock Switch Enable bit in Configuration Word 1 can be used to enable or disable the clock switching capability. When cleared, the NOSC and NDIV bits cannot be changed by user software. When set, writing to NOSC and NDIV is allowed and would switch the clock frequency.

### 7.3.1 NEW OSCILLATOR SOURCE (NOSC) AND NEW DIVIDER SELECTION REQUEST (NDIV) BITS

The New Oscillator Source (NOSC) and New Divider Selection Request (NDIV) bits of the OSCCON1 register select the system clock source and frequency that are used for the CPU and peripherals.

When new values of NOSC and NDIV are written to OSCCON1, the current oscillator selection will continue to operate while waiting for the new clock source to indicate that it is stable and ready. In some cases, the newly requested source may already be in use, and is ready immediately. In the case of a divider-only change, the new and old sources are the same, so the old source will be ready immediately. The device may enter Sleep while waiting for the switch as described in [Section 7.3.2 “Clock Switch and Sleep”](#).

When the new oscillator is ready, the New Oscillator Ready (NOSCR) bit of OSCCON3 and the Clock Switch Interrupt Flag (CSWIF) bit of the respective PIR register are set. If Clock Switch Interrupts are enabled (CSWIE = 1), an interrupt will be generated at that time. The Oscillator Ready (ORDY) bit of OSCCON3 can also be polled to determine when the oscillator is ready in lieu of an interrupt.

**Note:** The CSWIF interrupt will not wake the system from Sleep.

If the Clock Switch Hold (CSWHOLD) bit of OSCCON3 is clear, the oscillator switch will occur when the New Oscillator is Ready bit (NOSCR) is set, and the interrupt (if enabled) will be serviced at the new oscillator setting.

If CSWHOLD is set, the oscillator switch is suspended, while execution continues using the current (old) clock source. When the NOSCR bit is set, software should:

- Set CSWHOLD = 0 so the switch can complete, or
- Copy COSC into NOSC to abandon the switch.

If DOZE is in effect, the switch occurs on the next clock cycle, whether or not the CPU is operating during that cycle.

Changing the clock post-divider without changing the clock source (i.e., changing Fosc from 1 MHz to 2 MHz) is handled in the same manner as a clock source change, as described previously. The clock source will already be active, so the switch is relatively quick. CSWHOLD must be clear (CSWHOLD = 0) for the switch to complete.

The current COSC and CDIV are indicated in the OSCCON2 register up to the moment when the switch actually occurs, at which time OSCCON2 is updated and ORDY is set. NOSCR is cleared by hardware to indicate that the switch is complete.

## 7.3.2 CLOCK SWITCH AND SLEEP

If OSCCON1 is written with a new value and the device is put to Sleep before the switch completes, the switch will not take place and the device will enter Sleep mode.

When the device wakes from Sleep and the CSWHOLD bit is clear, the device will wake with the 'new' clock active, and the Clock Switch Interrupt flag bit (CSWIF) will be set.

When the device wakes from Sleep and the CSWHOLD bit is set, the device will wake with the 'old' clock active and the new clock will be requested again.

**FIGURE 7-6: CLOCK SWITCH (CSWHOLD = 0)**

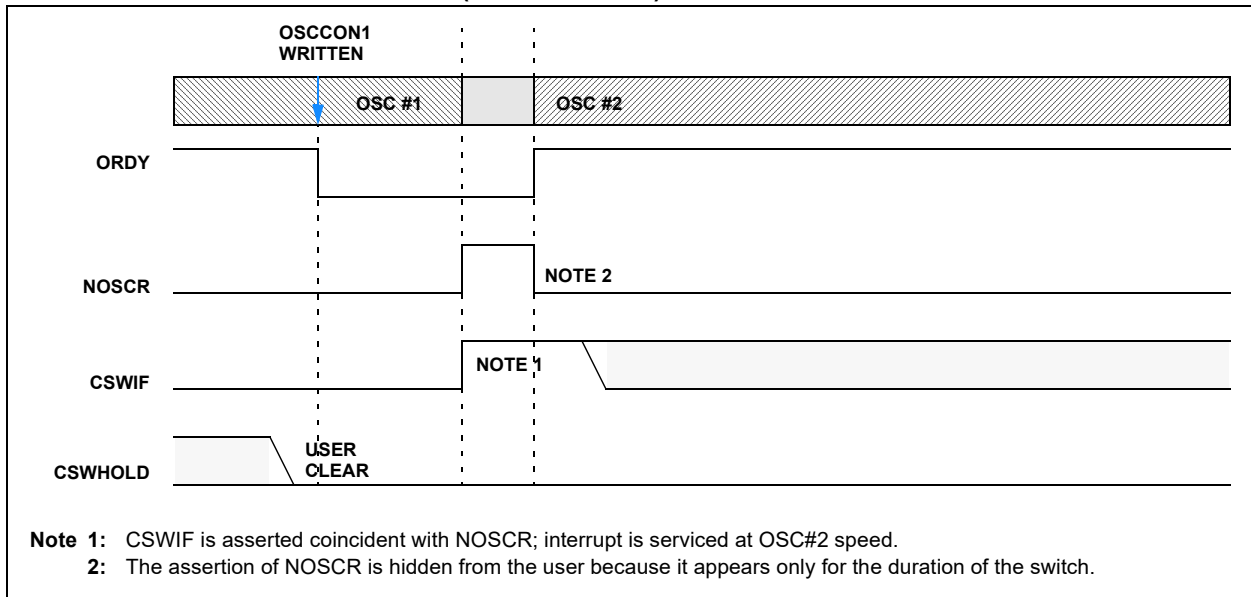


**FIGURE 7-7: CLOCK SWITCH (CSWHOLD = 1)**

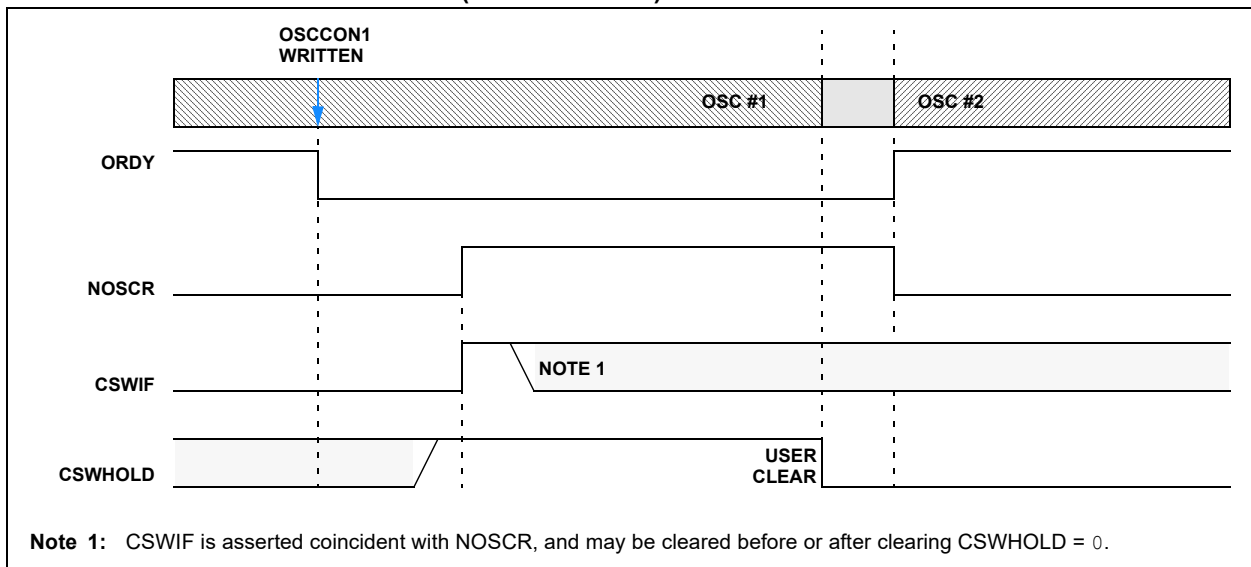
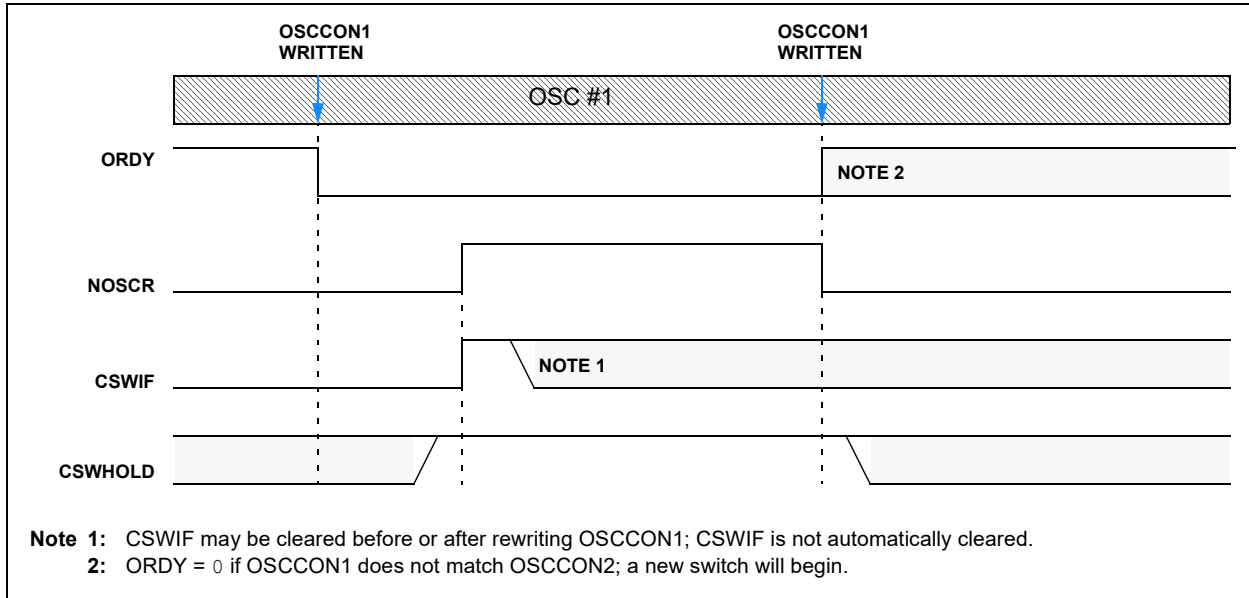


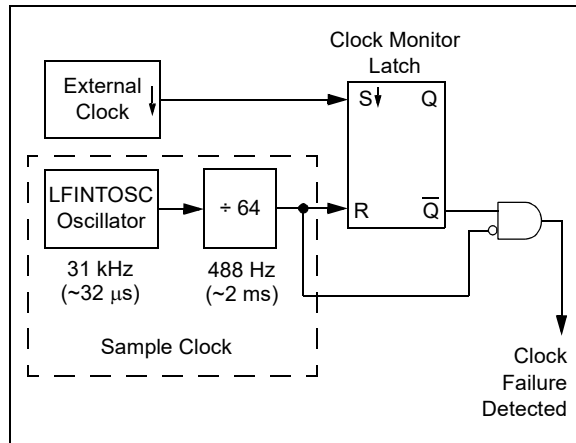
**FIGURE 7-8: CLOCK SWITCH ABANDONED**



## 7.4 Fail-Safe Clock Monitor

The Fail-Safe Clock Monitor (FSCM) allows the device to continue operating should the external oscillator fail. The FSCM is enabled by setting the FCMEN bit in the Configuration Words. The FSCM is applicable to all external Oscillator modes (LP, XT, HS, ECL/M/H and Secondary Oscillator).

**FIGURE 7-9: FSCM BLOCK DIAGRAM**



### 7.4.1 FAIL-SAFE DETECTION

The FSCM module detects a failed oscillator by comparing the external oscillator to the FSCM sample clock. The sample clock is generated by dividing the LFINTOSC by 64. See Figure 7-9. Inside the fail detector block is a latch. The external clock sets the latch on each falling edge of the external clock. The sample clock clears the latch on each rising edge of the sample clock. A failure is detected when an entire half-cycle of the sample clock elapses before the external clock goes low.

### 7.4.2 FAIL-SAFE OPERATION

When the external clock fails, the FSCM overwrites the COSC bits to select HFINTOSC (3'b110). The frequency of HFINTOSC would be determined by the previous state of the FRQ bits and the NDIV/CDIV bits. The bit flag OSFIF of the respective PIR register is set. Setting this flag will generate an interrupt if the OSFIE bit of the respective PIR register is also set. The device firmware can then take steps to mitigate the problems that may arise from a failed clock. The system clock will continue to be sourced from the internal clock source until the device firmware successfully restarts the external oscillator and switches back to external operation, by writing to the NOSC and NDIV bits of the OSCCON1 register.

### 7.4.3 FAIL-SAFE CONDITION CLEARING

The Fail-Safe condition is cleared after a Reset, executing a `SLEEP` instruction or changing the NOSC and NDIV bits of the OSCCON1 register. When switching to the external oscillator or PLL, the OST is restarted. While the OST is running, the device continues to operate from the INTOSC selected in OSCCON1. When the OST times out, the Fail-Safe condition is cleared after successfully switching to the external clock source. The OSCFIF bit should be cleared prior to switching to the external clock source. If the Fail-Safe condition still exists, the OSCFIF flag will again become set by hardware.

## 7.4.4 RESET OR WAKE-UP FROM SLEEP

The FSCM is designed to detect an oscillator failure after the Oscillator Start-up Timer (OST) has expired. The OST is used after waking up from Sleep and after any type of Reset. The OST is not used with the EC Clock modes so that the FSCM will be active as soon as the Reset or wake-up has completed.

**FIGURE 7-10: FSCM TIMING DIAGRAM**

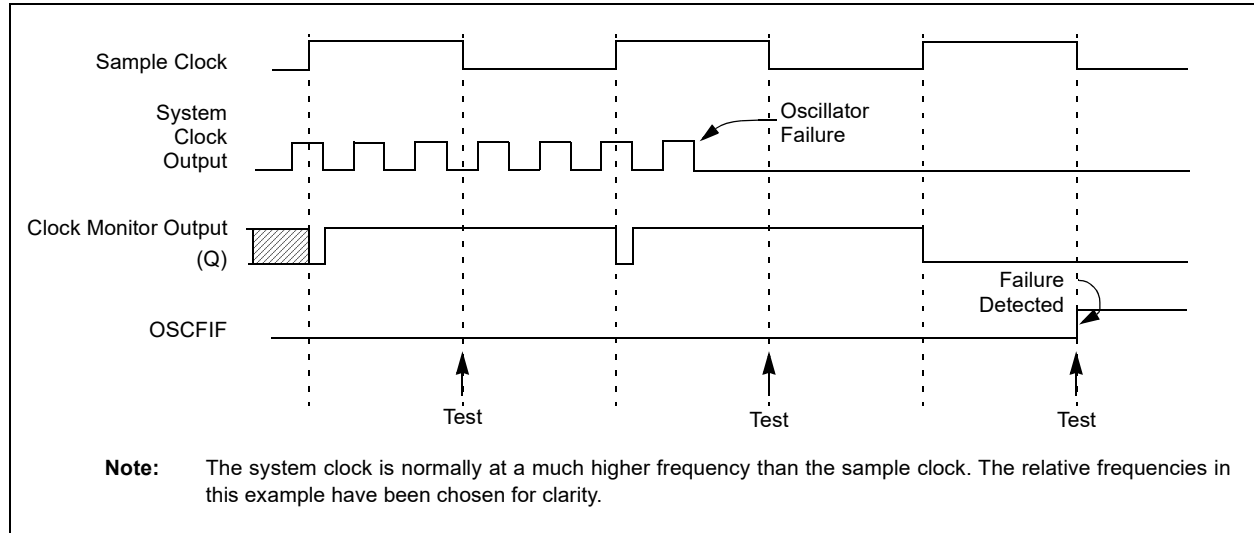


**TABLE 7-1: NOSC/COSC AND NDIV/CDIV BIT SETTINGS**

| NOSC<2:0><br>COSC<2:0> | Clock Source                   | NDIV<3:0><br>CDIV<3:0> | Clock Divider |
|------------------------|--------------------------------|------------------------|---------------|
| 111                    | EXTOSC <sup>(1)</sup>          | 1111-1010              | Reserved      |
| 110                    | HFINTOSC <sup>(2)</sup>        | 1001                   | 512           |
| 101                    | LFINTOSC                       | 1000                   | 256           |
| 100                    | SOSC                           | 0111                   | 128           |
| 011                    | Reserved                       | 0110                   | 64            |
| 010                    | EXTOSC + 4x PLL <sup>(3)</sup> | 0101                   | 32            |
| 001                    | Reserved                       | 0100                   | 16            |
| 000                    | Reserved                       | 0011                   | 8             |
|                        |                                | 0010                   | 4             |
|                        |                                | 0001                   | 2             |
|                        |                                | 0000                   | 1             |

- Note 1:** EXTOSC configured by the FEXTOSC bits of Configuration Word 1 ([Register 5-1](#)).
- Note 2:** HFINTOSC frequency is set with the FRQ bits of the OSCFRQ register ([Register 7-5](#)).
- Note 3:** EXTOSC must meet the PLL specifications ([Table 45-9](#)).

## 7.5 Register Definitions: Oscillator Control

### REGISTER 7-1: OSCCON1: OSCILLATOR CONTROL REGISTER 1

| U-0   | R/W-f/f   | R/W-f/f | R/W-f/f | R/W-q/q   | R/W-q/q | R/W-q/q | R/W-q/q |
|-------|-----------|---------|---------|-----------|---------|---------|---------|
| —     | NOSC<2:0> |         |         | NDIV<3:0> |         |         |         |
| bit 7 |           |         |         |           |         |         | bit 0   |

#### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared | f = determined by Configuration bit setting           |
|                      |                      | q = Reset value is determined by hardware             |

bit 7 **Unimplemented:** Read as '0'

bit 6-4 **NOSC<2:0>:** New Oscillator Source Request bits<sup>(1,2,3)</sup>  
The setting requests a source oscillator and PLL combination per [Table 7-1](#).  
POR value = RSTOSC ([Register 5-1](#)).

bit 3-0 **NDIV<3:0>:** New Divider Selection Request bits<sup>(2,3)</sup>  
The setting determines the new postscaler division ratio per [Table 7-1](#).

**Note 1:** The default value (f/f) is determined by the RSTOSC Configuration bits. See [Table 7-2](#) below.

**2:** If NOSC is written with a reserved value ([Table 7-1](#)), the operation is ignored and neither NOSC nor NDIV is written.

**3:** When CSWEN = 0, this register is read-only and cannot be changed from the POR value.

**TABLE 7-2: DEFAULT OSCILLATOR SETTINGS**

| RSTOSC | SFR Reset Values |      |        | Initial Fosc Frequency        |
|--------|------------------|------|--------|-------------------------------|
|        | NOSC/COSC        | CDIV | OSCFRQ |                               |
| 111    | 111              | 1:1  | 4 MHz  | EXTOSC per FEXTOSC            |
| 110    | 110              | 4:1  |        | Fosc = 1 MHz (4 MHz/4)        |
| 101    | 101              | 1:1  |        | LFINTOSC                      |
| 100    | 100              | 1:1  |        | SOSC                          |
| 011    | Reserved         |      |        |                               |
| 010    | 010              | 1:1  | 4 MHz  | EXTOSC + 4xPLL <sup>(1)</sup> |
| 001    | Reserved         |      |        |                               |
| 000    | 110              | 1:1  | 64 MHz | Fosc = 64 MHz                 |

**Note 1:** EXTOSC must meet the PLL specifications ([Table 45-9](#)).

## REGISTER 7-2: OSCCON2: OSCILLATOR CONTROL REGISTER 2

|       |           |       |       |           |       |       |       |
|-------|-----------|-------|-------|-----------|-------|-------|-------|
| U-0   | R-f/f     | R-f/f | R-f/f | R-f/f     | R-f/f | R-f/f | R-f/f |
| —     | COSC<2:0> |       |       | CDIV<3:0> |       |       |       |
| bit 7 |           |       |       |           |       |       | bit 0 |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7 **Unimplemented:** Read as '0'

bit 6-4 **COSC<2:0>:** Current Oscillator Source Select bits (read-only)<sup>(1)</sup>  
Indicates the current source oscillator and PLL combination per [Table 7-1](#).

bit 3-0 **CDIV<3:0>:** Current Divider Select bits (read-only)<sup>(1)</sup>  
Indicates the current postscaler division ratio per [Table 7-1](#).

**Note 1:** The POR value is the value present when user code execution begins.

## REGISTER 7-3: OSCCON3: OSCILLATOR CONTROL REGISTER 3

|            |         |     |       |       |     |     |       |
|------------|---------|-----|-------|-------|-----|-----|-------|
| R/W/HC-0/0 | R/W-0/0 | U-0 | R-0/0 | R-0/0 | U-0 | U-0 | U-0   |
| CSWHOLD    | SOSCPWR | —   | ORDY  | NOSCR | —   | —   | —     |
| bit 7      |         |     |       |       |     |     | bit 0 |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared | HC = Bit is cleared by hardware                       |

bit 7 **CSWHOLD:** Clock Switch Hold bit

- 1 = Clock switch will hold (with interrupt) when the oscillator selected by NOSC is ready
- 0 = Clock switch may proceed when the oscillator selected by NOSC is ready; NOSCR becomes '1', the switch will occur

bit 6 **SOSCPWR:** Secondary Oscillator Power Mode Select bit

- 1 = Secondary oscillator operating in High-Power mode
- 0 = Secondary oscillator operating in Low-Power mode

bit 5 **Unimplemented:** Read as '0'

bit 4 **ORDY:** Oscillator Ready bit (read-only)

- 1 = OSCCON1 = OSCCON2; the current system clock is the clock specified by NOSC
- 0 = A clock switch is in progress

bit 3 **NOSCR:** New Oscillator is Ready bit (read-only)<sup>(1)</sup>

- 1 = A clock switch is in progress and the oscillator selected by NOSC indicates a "ready" condition
- 0 = A clock switch is not in progress, or the NOSC-selected oscillator is not yet ready

bit 2-0 **Unimplemented:** Read as '0'

**Note 1:** If CSWHOLD = 0, the user may not see this bit set because, when the oscillator becomes ready there may be a delay of one instruction clock before this bit is set. The clock switch occurs in the next instruction cycle and this bit is cleared.

## REGISTER 7-4: OSCSTAT: OSCILLATOR STATUS REGISTER 1

| R-q/q | R-q/q | R-q/q | R-q/q | R-q/q | R-q/q | U-0 | R-q/q |
|-------|-------|-------|-------|-------|-------|-----|-------|
| EXTOR | HFOR  | MFOR  | LFOR  | SOR   | ADOR  | —   | PLL R |
| bit 7 |       |       |       |       |       |     | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

q = Reset value is determined by hardware

- bit 7 **EXTOR:** EXTOSC (external) Oscillator Ready bit  
1 = The oscillator is ready to be used  
0 = The oscillator is not enabled, or is not yet ready to be used
- bit 6 **HFOR:** HFINTOSC Oscillator Ready bit  
1 = The oscillator is ready to be used  
0 = The oscillator is not enabled, or is not yet ready to be used
- bit 5 **MFOR:** MFINTOSC Oscillator Ready bit  
1 = The oscillator is ready to be used  
0 = The oscillator is not enabled, or is not yet ready to be used
- bit 4 **LFOR:** LFINTOSC Oscillator Ready bit  
1 = The oscillator is ready to be used  
0 = The oscillator is not enabled, or is not yet ready to be used
- bit 3 **SOR:** Secondary (Timer1) Oscillator Ready bit  
1 = The oscillator is ready to be used  
0 = The oscillator is not enabled, or is not yet ready to be used
- bit 2 **ADOR:** ADC Oscillator Ready bit  
1 = The oscillator is ready to be used  
0 = The oscillator is not enabled, or is not yet ready to be used
- bit 1 **Unimplemented:** Read as '0'
- bit 0 **PLL R:** PLL is Ready bit  
1 = The PLL is ready to be used  
0 = The PLL is not enabled, the required input source is not ready, or the PLL is not locked.



## REGISTER 7-5: OSCFRQ: HFINTOSC FREQUENCY SELECTION REGISTER

|       |     |     |     |          |         |         |         |
|-------|-----|-----|-----|----------|---------|---------|---------|
| U-0   | U-0 | U-0 | U-0 | R/W-q/q  | R/W-q/q | R/W-q/q | R/W-q/q |
| —     | —   | —   | —   | FRQ<3:0> |         |         |         |
| bit 7 |     |     |     | bit 0    |         |         |         |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

q = Reset value is determined by hardware

bit 7-4

**Unimplemented:** Read as '0'

bit 3-0

**FRQ<3:0>:** HFINTOSC Frequency Selection bits<sup>(1)</sup>

| FRQ<3:0> | Nominal Freq (MHz) |
|----------|--------------------|
| 1001     | Reserved           |
| 1010     |                    |
| 1111     |                    |
| 1110     |                    |
| 1101     |                    |
| 1100     |                    |
| 1011     |                    |
| 1000     | 64                 |
| 0111     | 48                 |
| 0110     | 32                 |
| 0101     | 16                 |
| 0100     | 12                 |
| 0011     | 8                  |
| 0010     | 4                  |
| 0001     | 2                  |
| 0000     | 1                  |

**Note 1:** Refer to [Table 7-2](#) for more information.

## REGISTER 7-6: OSCTUNE: HFINTOSC TUNING REGISTER

| U-0   | U-0 | R/W-0/0  | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|-------|-----|----------|---------|---------|---------|---------|---------|
| —     | —   | TUN<5:0> |         |         |         |         |         |
| bit 7 |     | bit 0    |         |         |         |         |         |

## REGISTER 7-7: OSCEN: OSCILLATOR MANUAL ENABLE REGISTER

|         |         |         |         |         |         |     |     |
|---------|---------|---------|---------|---------|---------|-----|-----|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | U-0 | U-0 |
| EXTOEN  | HFOEN   | MFOEN   | LFOEN   | SOSCEN  | ADOEN   | —   | —   |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

- bit 7      **EXTOEN:** External Oscillator Manual Request Enable bit  
1 = EXTOSC is explicitly enabled, operating as specified by FEXTOSC  
0 = EXTOSC could be enabled by requesting peripheral
- bit 6      **HFOEN:** HFINTOSC Oscillator Manual Request Enable bit  
1 = HFINTOSC is explicitly enabled, operating as specified by OSCFRQ ([Register 7-5](#))  
0 = HFINTOSC could be enabled by requesting peripheral
- bit 5      **MFOEN:** MFINTOSC (500 kHz/31.25 kHz) Oscillator Manual Request Enable bit (Derived from HFINTOSC)  
1 = MFINTOSC is explicitly enabled  
0 = MFINTOSC could be enabled by requesting peripheral
- bit 4      **LFOEN:** LFINTOSC (31 kHz) Oscillator Manual Request Enable bit  
1 = LFINTOSC is explicitly enabled  
0 = LFINTOSC could be enabled by requesting peripheral
- bit 3      **SOSCEN:** Secondary Oscillator Manual Request Enable bit  
1 = Secondary Oscillator is explicitly enabled, operating as specified by SOSCPWR  
0 = Secondary Oscillator could be enabled by requesting peripheral
- bit 2      **ADOEN:** ADC Oscillator Manual Request Enable bit  
1 = ADC oscillator is explicitly enabled  
0 = ADC oscillator could be enabled by requesting peripheral
- bit 1-0    **Unimplemented:** Read as '0'

**TABLE 7-3: SUMMARY OF REGISTERS ASSOCIATED WITH CLOCK SOURCES**

| Name    | Bit 7   | Bit 6     | Bit 5    | Bit 4 | Bit 3     | Bit 2 | Bit 1 | Bit 0 | Register on Page   |
|---------|---------|-----------|----------|-------|-----------|-------|-------|-------|--------------------|
| OSCCON1 | —       | NOSC<2:0> |          |       | NDIV<3:0> |       |       |       | <a href="#">94</a> |
| OSCCON2 | —       | COSC<2:0> |          |       | CDIV<3:0> |       |       |       | <a href="#">95</a> |
| OSCCON3 | CSWHOLD | SOSCPWR   | —        | ORDY  | NOSCR     | —     | —     | —     | <a href="#">95</a> |
| OSCSTAT | EXTOR   | HFOR      | MFOR     | LFOR  | SOR       | ADOR  | —     | PLLRL | <a href="#">96</a> |
| OSCTUNE | —       | —         | TUN<5:0> |       |           |       |       |       | <a href="#">98</a> |
| OSCFRQ  | —       | —         | —        | —     | FRQ<3:0>  |       |       |       | <a href="#">97</a> |
| OSCEN   | EXTOEN  | HFOEN     | MFOEN    | LFOEN | SOSCEN    | ADOEN | —     | —     | <a href="#">99</a> |

**Legend:** — = unimplemented location, read as '0'. Shaded cells are not used by clock sources.

## 8.0 REFERENCE CLOCK OUTPUT MODULE

The reference clock output module provides the ability to send a clock signal to the clock reference output pin (CLKR). The reference clock output can also be used as a signal for other peripherals, such as the Data Signal Modulator (DSM), Memory Scanner and Timer module.

The reference clock output module has the following features:

- Selectable clock source using the CLKRCLK register
- Programmable clock divider
- Selectable duty cycle

**FIGURE 8-1: CLOCK REFERENCE BLOCK DIAGRAM**

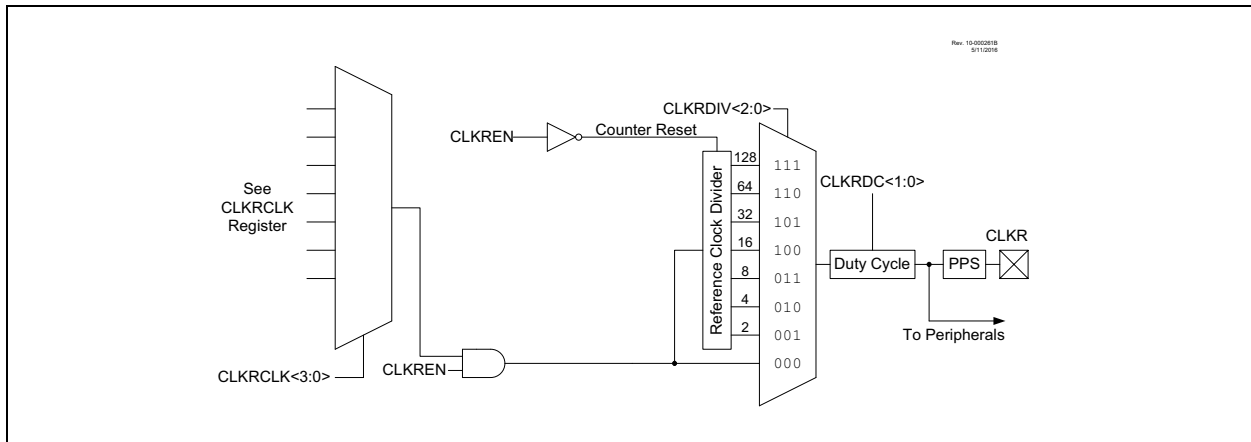
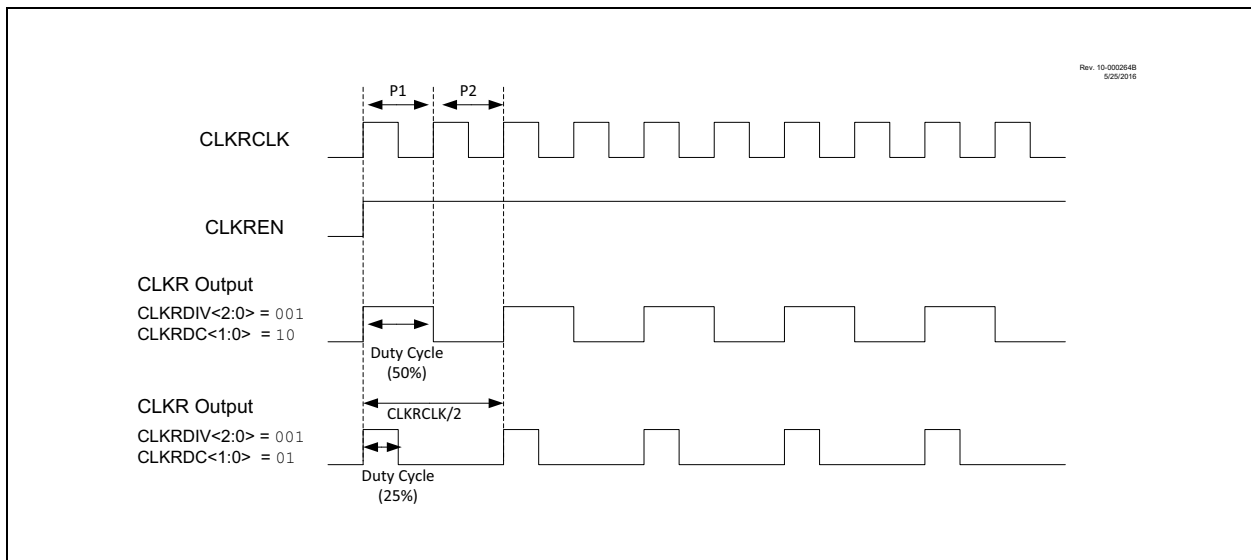


**FIGURE 8-2: CLOCK REFERENCE TIMING**



## 8.1 Clock Source

The input to the reference clock output can be selected using the CLKRCLK register.

### 8.1.1 CLOCK SYNCHRONIZATION

Once the reference clock enable (EN) is set, the module is ensured to be glitch-free at start-up.

When the reference clock output is disabled, the output signal will be disabled immediately.

Clock dividers and clock duty cycles can be changed while the module is enabled, but glitches may occur on the output. To avoid possible glitches, clock dividers and clock duty cycles should be changed only when the CLKREN is clear.

## 8.2 Programmable Clock Divider

The module takes the clock input and divides it based on the value of the DIV<2:0> bits of the CLKRCON register ([Register 8-1](#)).

The following configurations can be made based on the DIV<2:0> bits:

- Base FOSC value
- FOSC divided by 2
- FOSC divided by 4
- FOSC divided by 8
- FOSC divided by 16
- FOSC divided by 32
- FOSC divided by 64
- FOSC divided by 128

The clock divider values can be changed while the module is enabled; however, in order to prevent glitches on the output, the DIV<2:0> bits should only be changed when the module is disabled (EN = 0).

## 8.3 Selectable Duty Cycle

The DC<1:0> bits of the CLKRCON register can be used to modify the duty cycle of the output clock. A duty cycle of 25%, 50%, or 75% can be selected for all clock rates, with the exception of the undivided base FOSC value.

The duty cycle can be changed while the module is enabled; however, in order to prevent glitches on the output, the DC<1:0> bits should only be changed when the module is disabled (EN = 0).

|                                                                                             |
|---------------------------------------------------------------------------------------------|
| <b>Note:</b> The DC1 bit is reset to '1'. This makes the default duty cycle 50% and not 0%. |
|---------------------------------------------------------------------------------------------|

## 8.4 Operation in Sleep Mode

The reference clock output module clock is based on the system clock. When the device goes to Sleep, the module outputs will remain in their current state. This will have a direct effect on peripherals using the reference clock output as an input signal. No change should occur in the module from entering or exiting from Sleep.

## 8.5 Register Definitions: Reference Clock

Long bit name prefixes for the Reference Clock peripherals are shown below. Refer to [Section 1.3.2.2 “Long Bit Names”](#) for more information.

| Peripheral | Bit Name Prefix |
|------------|-----------------|
| CLKR       | CLKR            |

### REGISTER 8-1: CLKRCON: REFERENCE CLOCK CONTROL REGISTER

|         |     |     |         |         |         |          |         |
|---------|-----|-----|---------|---------|---------|----------|---------|
| R/W-0/0 | U-0 | U-0 | R/W-1/1 | R/W-0/0 | R/W-0/0 | R/W-0/0  | R/W-0/0 |
| EN      | —   | —   | DC<1:0> |         |         | DIV<2:0> |         |
| bit 7   |     |     |         |         |         |          | bit 0   |

#### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7 **EN:** Reference Clock Module Enable bit  
 1 = Reference clock module enabled  
 0 = Reference clock module is disabled

bit 6-5 **Unimplemented:** Read as '0'

bit 4-3 **DC<1:0>:** Reference Clock Duty Cycle bits<sup>(1)</sup>  
 11 = Clock outputs duty cycle of 75%  
 10 = Clock outputs duty cycle of 50%  
 01 = Clock outputs duty cycle of 25%  
 00 = Clock outputs duty cycle of 0%

bit 2-0 **DIV<2:0>:** Reference Clock Divider bits  
 111 = Base clock value divided by 128  
 110 = Base clock value divided by 64  
 101 = Base clock value divided by 32  
 100 = Base clock value divided by 16  
 011 = Base clock value divided by 8  
 010 = Base clock value divided by 4  
 001 = Base clock value divided by 2  
 000 = Base clock value

**Note 1:** Bits are valid for reference clock divider values of two or larger, the base clock cannot be further divided.

# PIC18(L)F25/26K83

**REGISTER 8-2: CLKRCLK: CLOCK REFERENCE CLOCK SELECTION MUX**

|       |     |     |     |          |         |         |         |
|-------|-----|-----|-----|----------|---------|---------|---------|
| U-0   | U-0 | U-0 | U-0 | R/W-0/0  | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| —     | —   | —   | —   | CLK<3:0> |         |         |         |
| bit 7 |     |     |     | bit 0    |         |         |         |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-4 **Unimplemented:** Read as '0'

bit 3-0 **CLK<3:0>:** CLKR Clock Selection bits

1111 = Reserved

- 
- 
- 

1011 = Reserved

1010 = CLC4 Output

1001 = CLC3 Output

1000 = CLC2 Output

0111 = CLC1 Output

0110 = NCO1 Output

0101 = SOSC

0100 = MFINTOSC (31.25 kHz)

0011 = MFINTOSC (500 kHz)

0010 = LFINTOSC (31 kHz)

0001 = HFINTOSC

0000 = Fosc

**TABLE 8-1: SUMMARY OF REGISTERS ASSOCIATED WITH CLOCK REFERENCE OUTPUT**

| Name    | Bit 7 | Bit 6 | Bit 5 | Bit 4   | Bit 3 | Bit 2    | Bit 1 | Bit 0 | Register on Page    |
|---------|-------|-------|-------|---------|-------|----------|-------|-------|---------------------|
| CLKRCON | EN    | —     | —     | DC<1:0> |       | DIV<2:0> |       |       | <a href="#">103</a> |
| CLKRCLK | —     | —     | —     | —       | —     | CLK<2:0> |       |       | <a href="#">104</a> |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by the CLKR module.



## 9.0 INTERRUPT CONTROLLER

The vectored interrupt controller module reduces the numerous peripheral interrupt request signals to a single interrupt request signal to the CPU. This module includes the following major features:

- Interrupt Vector Table (IVT) with a unique vector for each interrupt source
- Fixed and ensured interrupt latency
- Programmable base address for Interrupt Vector Table (IVT) with lock
- Two user-selectable priority levels – High priority and Low priority
- Two levels of context saving
- Interrupt state Status bits to indicate the current execution status of the CPU

The interrupt controller module assembles all of the interrupt request signals and resolves the interrupts based on both a fixed natural order priority (i.e., determined by the Interrupt Vector Table), and a user-assigned priority (i.e., determined by the IPRx registers), thereby eliminating scanning of interrupt sources.

### 9.1 Interrupt Control and Status Registers

The devices in this family implement the following registers for the interrupt controller:

- INTCON0, INTCON1 Control Registers
- PIRx – Peripheral Interrupt Status Registers
- PIEx – Peripheral Interrupt Enable Registers
- IPRx – Peripheral Interrupt Priority Registers
- IVTBASE<20:0> Address Registers
- IVTLOCK Register

Global interrupt control functions and external interrupts are controlled from the INTCON0 register. The INTCON1 register contains the status flags for the Interrupt controller.

The PIRx registers contain all of the interrupt request flags. Each source of interrupt has a Status bit, which is set by the respective peripherals or an external signal and is cleared via software.

The PIEx registers contain all of the interrupt enable bits. These control bits are used to individually enable interrupts from the peripherals or external signals.

The IPRx registers are used to set the Interrupt Priority Level for each source of interrupt. Each user interrupt source can be assigned to either a high or low priority.

The IVTBASE register is user programmable and is used to determine the start address of the Interrupt Vector Table and the IVTLOCK register is used to prevent any unintended writes to the IVTBASE register.

There are two other Configuration bits that control the way the interrupt controller can be configured.

- CONFIG2L<3>, MVECEN bit
- CONFIG2L<4>, IVT1WAY bit

The MVECEN bit in CONFIG2L determines whether the Vector table is used to determine the interrupt priorities.

- When the IVT1WAY determines the number of times the IVTLOCKED bit can be cleared and set after a device Reset. See [Section 9.2.3 “Interrupt Vector Table \(IVT\) address calculation”](#) for details.

## 9.2 Interrupt Vector Table (IVT)

The interrupt controller supports an Interrupt Vector Table (IVT) that contains the vector address location for each interrupt request source.

The Interrupt Vector Table (IVT) resides in program memory, starting at address location determined by the IVTBASE registers; refer to Registers 9-33 through 9-35 for details. The IVT contains 68 vectors, one for each source of interrupt. Each interrupt vector location contains the starting address of the associated Interrupt Service Routine (ISR).

The MVECEN bit in Configuration Word 2L controls the availability of the vector table.

### 9.2.1 INTERRUPT VECTOR TABLE BASE ADDRESS (IVTBASE)

The start address of the vector table is user programmable through the IVTBASE registers. The user must ensure the start address is such that it can encompass the entire vector table inside the program memory.

Each vector address is a 16-bit word (or two address locations on PIC18 devices). So for  $n$  interrupt sources, there are  $2n$  address locations necessary to hold the table starting from IVTBASE as the first location. So the starting address of IVTBASE should be chosen such that the address range from IVTBASE to  $(IVTBASE + 2n - 1)$  can be encompassed inside the program flash memory.

For example, the PIC18F26K83 devices have the highest vector number: 81. So IVTBASE should be chosen such that  $(IVTBASE + 0xA1)$  is less than the last memory location in program flash memory.

A programmable vector table base address is useful in situations to switch between different sets of vector tables, depending on the application. It can also be used when the application program needs to update the existing vector table (vector address values).

**Note:** It is required that the user assign an even address to the IVTBASE register for correct operation.

### 9.2.2 INTERRUPT VECTOR TABLE CONTENTS

#### MVECEN = 0

When  $MVECEN = 0$ , the address location pointed by the IVTBASE registers has a GOTO instruction for a high priority interrupt. Similarly, the corresponding low priority vector location also has a GOTO instruction, which is executed in case of a low priority interrupt.

#### MVECEN = 1

When  $MVECEN = 1$ , the value in the vector table of each interrupt, points to the address location of the first instruction of the interrupt service routine.

ISR Location = Interrupt Vector Table entry  $\ll 2$ .

### 9.2.3 INTERRUPT VECTOR TABLE (IVT) ADDRESS CALCULATION

#### MVECEN = 0

When the MVECEN bit in Configuration Word 2L (Register 5-3) is cleared, the address pointed by IVTBASE registers is used as the high priority interrupt vector address. The low priority interrupt vector address is offset eight instruction words from the address in IVTBASE registers.

For PIC18 devices the IVTBASE registers default to 00 0008h, the high priority interrupt vector address will be 00 0008h and the low priority interrupt vector address will be 00 0018h.

#### MVECEN = 1

Each interrupt has a unique vector number associated with it as defined in Table 9-2. This vector number is used for calculating the location of the interrupt vector for a particular interrupt source.

Interrupt Vector Address =  $IVTBASE + (2 * \text{Vector Number})$ .

This calculated Interrupt Vector Address value is stored in the IVTAD<20:0> registers when an interrupt is received (Registers 9-36 through 9-38).

User-assigned software priority assigned using the IPRx registers does not affect address calculation and is only used to resolve concurrent interrupts.

If for any reason the address of the ISR could not be fetched from the vector table, it will cause the system to reset and clear the memory execution violation flag (MEMV bit) in PCON1 register (Register 6-3). This occurs due to any one of the following:

- The entry for the interrupt in the vector table lies outside the executable PFM area (SAF area is non-executable when  $SAFEN = 1$ ).
- ISR pointed by the vector table lies outside the executable PFM area (SAF area is non-executable when  $SAFEN = 1$ ).

**TABLE 9-1: IVT ADDRESS CALCULATION SUMMARY**

| IVT Address Calculation                                 |   | Interrupt Priority<br>INTCON0 Register, IPEN bit |                                   |
|---------------------------------------------------------|---|--------------------------------------------------|-----------------------------------|
|                                                         |   | 0                                                | 1                                 |
| Multi-Vector Enable<br>CONFIG 2L register<br>MVECEN bit | 0 | IVTBASE                                          | High Priority<br>IVTBASE          |
|                                                         |   |                                                  | Low Priority<br>IVTBASE + 8 words |
|                                                         | 1 | IVTBASE + 2*(Vector Number)                      |                                   |

## 9.2.4 ACCESS CONTROL FOR IVTBASE REGISTERS

The Interrupt controller has an IVTLOCKED bit which can be set to avoid inadvertent changes to the IVTBASE registers contents. Setting and clearing this bit requires a special sequence as an extra precaution against inadvertent changes.

To allow writes to IVTBASE registers, the interrupts must be disabled (GIEH = 0) and the IVTLOCKED bit must be cleared. The user must follow the sequence shown in [Example 9-1](#) to clear the IVTLOCKED bit.

### EXAMPLE 9-1: IVT UNLOCK SEQUENCE

```
; Disable Interrupts:
    BCF          INTCON0, GIE;
; Bank to IVTLOCK register
    BANKSEL      IVTLOCK;
    MOVLW        55h;

; Required sequence, next 4 instructions
    MOVWF        IVTLOCK;
    MOVLW        AAh;
    MOVWF        IVTLOCK;

; Clear IVTLOCKED bit to enable writes
    BCF          IVTLOCK, IVTLOCKED;

; Enable Interrupts
    BSF          INTCON0, GIE;
```

The user must follow the sequence shown in [Example 9-2](#) to set the IVTLOCKED bit.

### EXAMPLE 9-2: IVT LOCK SEQUENCE

```
; Disable Interrupts:
    BCF          INTCON0, GIE;
; Bank to IVTLOCK register
    BANKSEL      IVTLOCK;
    MOVLW        55h;

; Required sequence, next 4 instructions
    MOVWF        IVTLOCK;
    MOVLW        AAh;
    MOVWF        IVTLOCK;

; Set IVTLOCKED bit to enable writes
    BSF          IVTLOCK, IVTLOCKED;

; Enable Interrupts
    BSF          INTCON0, GIE;
```

When the IVT1WAY Configuration bit is set, the IVTLOCKED bit can be cleared and set only once after a device Reset. The unlock operation in [Example 9-1](#) will have no effect after the lock sequence in [Example 9-2](#) is used to set the IVTLOCK. Unlocking is inhibited until a system Reset occurs.

## 9.3 Interrupt Priority

The final priority level for any pending source of interrupt is determined first by the user-assigned priority of that source in the IPRx register, then by the natural order priority within the IVT. The sections below detail the operation of Interrupt priorities.

### 9.3.1 USER (SOFTWARE) PRIORITY

User-assigned interrupt priority is enabled by setting the IPEN bit in the INTCON0 register ([Register 9-1](#)). Each peripheral interrupt source can be assigned a high or low priority level by the user. The user-assignable interrupt priority control bits for each interrupt are located in the IPRx registers ([Registers 9-23 through 9-32](#)).

The interrupts are serviced based on predefined interrupt priority scheme defined below.

1. Interrupts set by the user as high-priority interrupt have higher precedence of execution. High-priority interrupts will override a low-priority request when:
  - a) A low priority interrupt has been requested or its request is already pending.
  - b) A low- and high-priority interrupt are triggered concurrently, i.e., on the same instruction cycle<sup>(1)</sup>.
  - c) A low-priority interrupt was requested and the corresponding Interrupt Service Routine is currently executing. In this case, the lower priority interrupt routine will complete executing after the high-priority interrupt has been serviced<sup>(2)</sup>.
2. Interrupts set by the user as a low priority have the lower priority of execution and are preempted by any high-priority interrupt.
3. Interrupts defined with the same software priority cannot preempt or interrupt each other. Concurrent pending interrupts with the same user priority are resolved using the natural order priority. (when MVECEN = ON) or in the order the interrupt flag bits are polled in the ISR (when MVECEN = OFF).

**Note 1:** When a high priority interrupt preempts a concurrent low priority interrupt, the GIEL bit may be cleared in the high priority Interrupt Service Routine. If the GIEL bit is cleared, the low priority interrupt will NOT be serviced even if it was originally requested. The corresponding interrupt flag needs to be cleared in user code.

**2:** When a high priority interrupt is requested while a low priority Interrupt Service Routine is executing, the GIEL bit may be cleared in the high priority Interrupt Service Routine. The pending low priority interrupt will resume even if the GIEL bit is cleared.

## 9.3.2 NATURAL ORDER (HARDWARE) PRIORITY

When more than one interrupt with the same user specified priority level are requested, the priority conflict is resolved by using a method called “Natural Order Priority”. Natural order priority is a fixed priority scheme that is based on the Interrupt Vector Table. [Table 9-2](#) shows the natural order priority and the interrupt vector number assigned for each source.

**TABLE 9-2: INTERRUPT VECTOR PRIORITY TABLE**

| Vector Number | Interrupt Source   | Vector Number | Interrupt Source |
|---------------|--------------------|---------------|------------------|
| 0             | Software Interrupt | 42            | TXB0IF           |
| 1             | HLVD               | 43            | TXB1IF           |
| 2             | OSF                | 44            | TXB2IF/TXBnIF    |
| 3             | CSW                | 45            | ERRIF            |
| 4             | NVM                | 46            | WAKIF            |
| 5             | SCAN               | 47            | IRXIF            |
| 6             | CRC                | 48            | C2               |
| 7             | IOC                | 49            | SMT2             |
| 8             | INT0               | 50            | SMT2PRA          |
| 9             | ZCD                | 51            | SMT2PWA          |
| 10            | AD                 | 52            | DMA2SCNT         |
| 11            | ADT                | 53            | DMA2DCNT         |
| 12            | C1                 | 54            | DMA2OR           |
| 13            | SMT1               | 55            | DMA2A            |
| 14            | SMT1PRA            | 56            | I2C2RX           |
| 15            | SMT1PWA            | 57            | I2C2TX           |
| 16            | DMA1SCNT           | 58            | I2C2             |
| 17            | DMA1DCNT           | 59            | I2C2E            |
| 18            | DMA1OR             | 60            | U2RX             |
| 19            | DMA1A              | 61            | U2TX             |
| 20            | SPI1RX             | 62            | U2E              |
| 21            | SPI1TX             | 63            | U2               |
| 22            | SPI1               | 64            | TMR3             |
| 23            | I2C1RX             | 65            | TMR3G            |
| 24            | I2C1TX             | 66            | TMR4             |
| 25            | I2C1               | 67            | CCP2             |
| 26            | I2C1E              | 68            | CWG2             |
| 27            | U1RX               | 69            | CLC2             |
| 28            | U1TX               | 70            | INT2             |
| 29            | U1E                | 71            | TMR5             |
| 30            | U1                 | 72            | TMR5G            |
| 31            | TMR0               | 73            | TMR6             |
| 32            | TMR1               | 74            | CCP3             |
| 33            | TMR1G              | 75            | CWG3             |
| 34            | TMR2               | 76            | CLC3             |
| 35            | CCP1               | 77            | CCP4             |
| 36            | NCO                | 78            | CLC4             |
| 37            | CWG1               | 79            | —                |
| 38            | CLC1               | 80            | —                |
| 39            | INT1               | 81            | —                |
| 40            | RXB0IF/FIFOIF      |               |                  |
| 41            | RXB1IF/RXBnIF      |               |                  |

The natural order priority scheme has vector interrupt 0 as the highest priority and vector interrupt 81 as the lowest priority.

For example, when two concurrently occurring interrupt sources that are both designated high priority using the IPRx register will be resolved using the natural order priority (i.e., the interrupt with a lower corresponding vector number will preempt the interrupt with the higher vector number).

The ability for the user to assign every interrupt source to high or low priority levels means that the user program can give an interrupt with a low natural order priority a higher overall priority level.

## 9.4 Interrupt Operation

All pending interrupts are indicated by the flag bit being equal to a ‘1’ in the PIRx register. All pending interrupts are resolved using the priority scheme explained in [Section 9.3 “Interrupt Priority”](#).

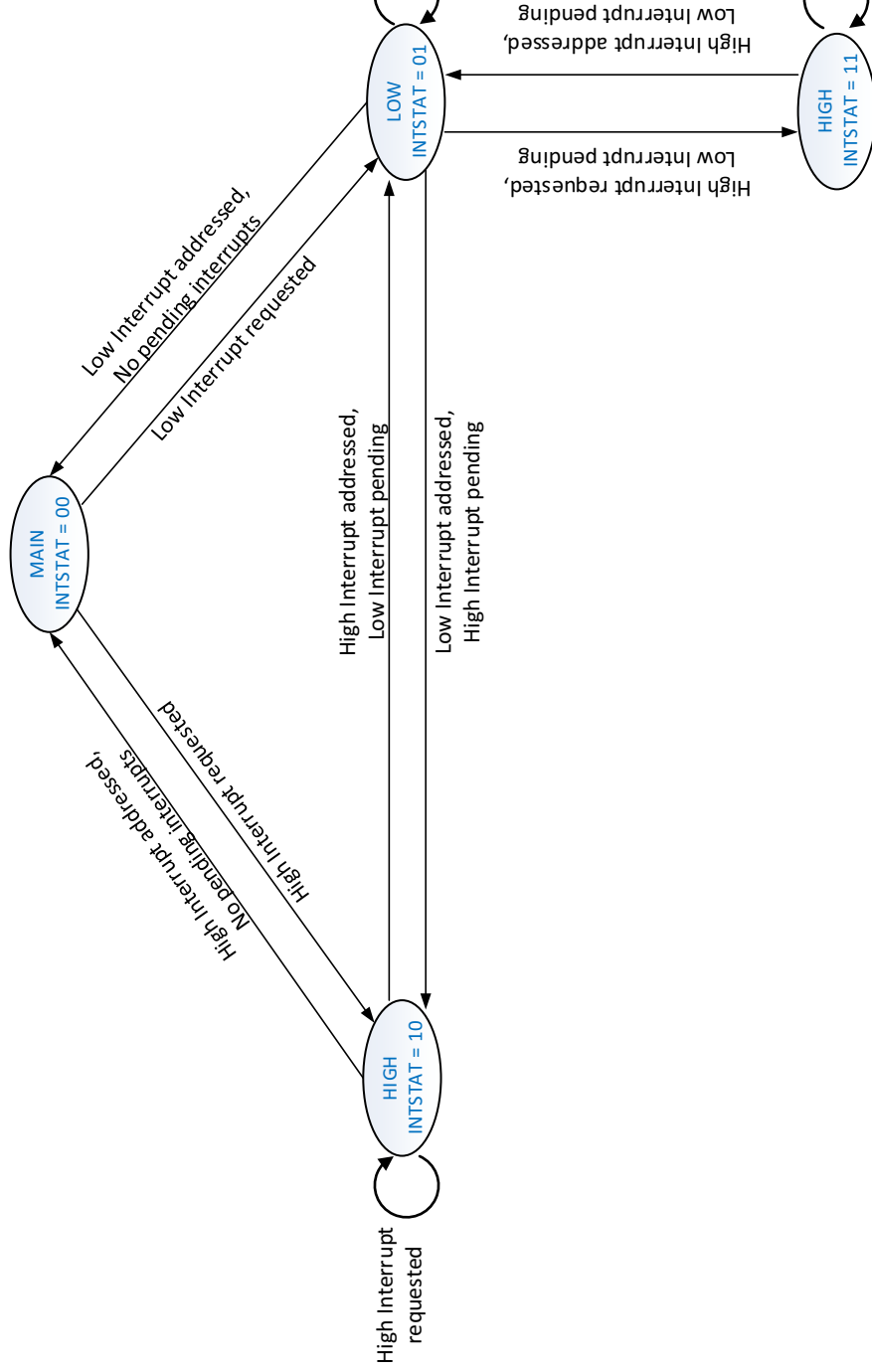
Once the interrupt source to be serviced is resolved, the program execution vectors to the resolved interrupt vector addresses, as explained in [Section 9.2 “Interrupt Vector Table \(IVT\)”](#). The vector number is also stored in the WREG register. Most of the flag bits are required to be cleared by the application software, but in some cases, device hardware clears the interrupt automatically. Some flag bits are read-only in the PIRx registers, these flags are a summary of the source interrupts and the corresponding interrupt flags of the source must be cleared.

A valid interrupt can be either a high or low priority interrupt when in main routine or a high priority interrupt when in low priority Interrupt Service Routine. Depending on order of interrupt requests received and their relative timing, the CPU will be in the state of execution indicated by the STAT bits of the INTCON1 register ([Register 9-2](#)).

The State machine shown in [Figure 9-1](#) and the subsequent sections detail the execution of interrupts when received in different orders.

**Note:** The state of GIEH/L is not changed by the hardware when servicing an interrupt. The internal state machine is used to keep track of execution states. These bits can be manipulated in the user code resulting in transferring execution to the main routine and ignoring existing interrupts.

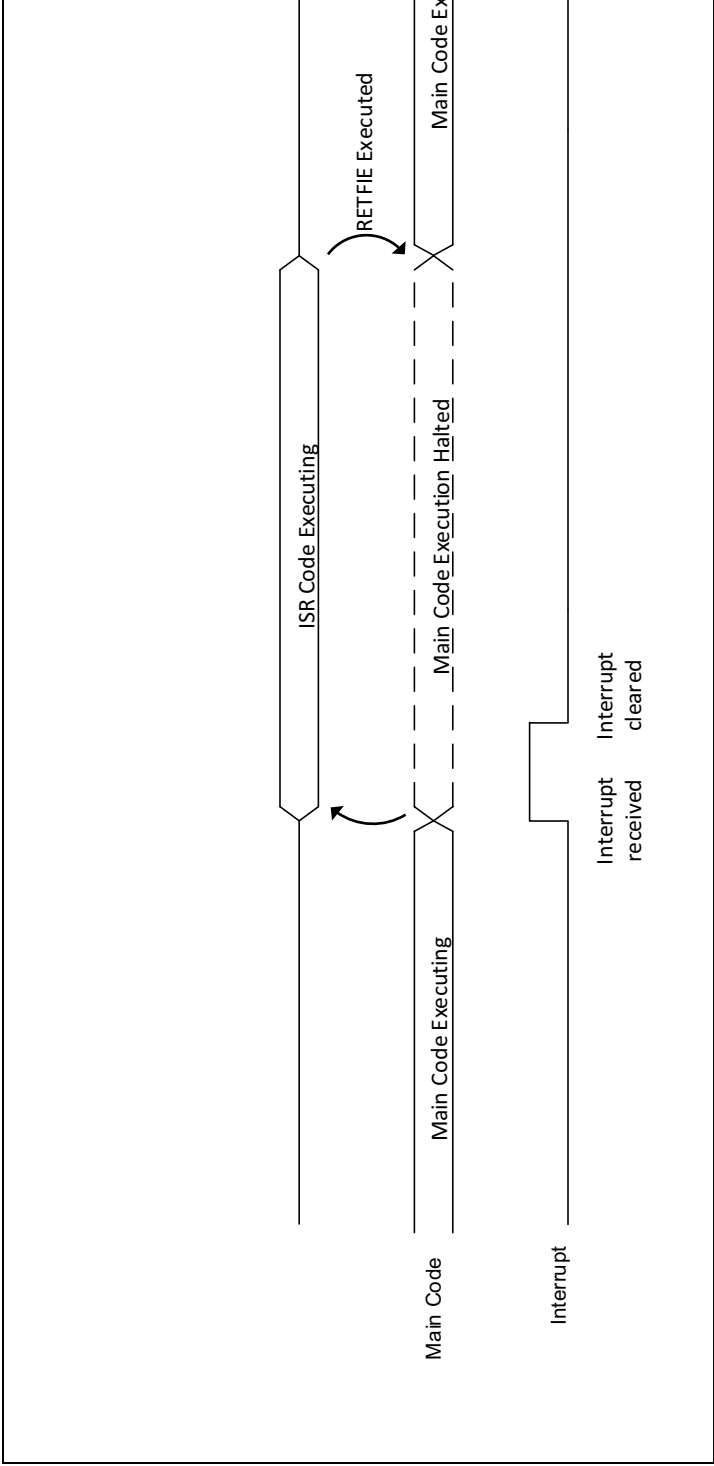
**FIGURE 9-1: VECTORED INTERRUPTS STATE TRANSITION DIAGRAM**



9.4.1 SERVING A HIGH OR LOW PRIORITY INTERRUPT  
WHEN MAIN ROUTINE CODE IS EXECUTING

When a high or low priority interrupt is requested when the main routine code is executing, the main routine execution is halted and the ISR is addressed, see [Figure 9-2](#). Upon a return from the ISR (by executing the `RETFIE` instruction), the main routine resumes execution.

**FIGURE 9-2: INTERRUPT EXECUTION: HIGH/LOW PRIORITY INTERRUPT WHEN EXECUTING MAIN ROUTINE**

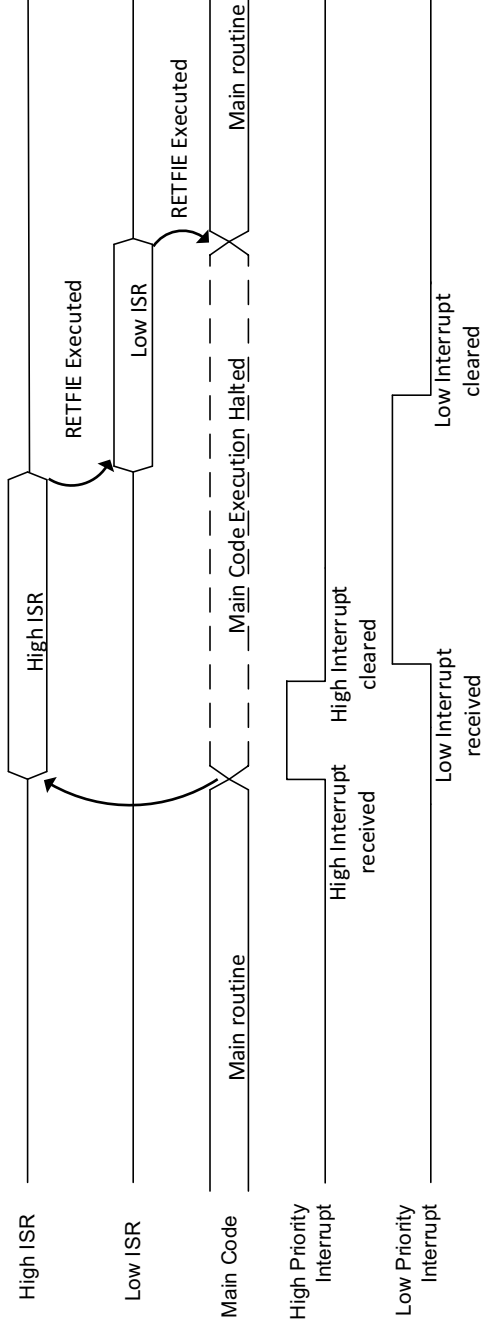


#### 9.4.2 SERVING A HIGH PRIORITY INTERRUPT WHILE A LOW PRIORITY INTERRUPT PENDING

A high priority interrupt request will always take precedence over any interrupt of a lower priority. The high priority interrupt is acknowledged first, then the low-priority interrupt is acknowledged. Upon a return from the high priority ISR (by executing the `RETFIE` instruction), the low priority interrupt is serviced, see [Figure 9-3](#).

If any other high priority interrupts are pending and enabled, then they are serviced before servicing the pending low priority interrupt. If no other high priority interrupt requests are active, the low priority interrupt is serviced.

**FIGURE 9-3: INTERRUPT EXECUTION: HIGH PRIORITY INTERRUPT WITH A LOW PRIORITY INTERRUPT PENDING**



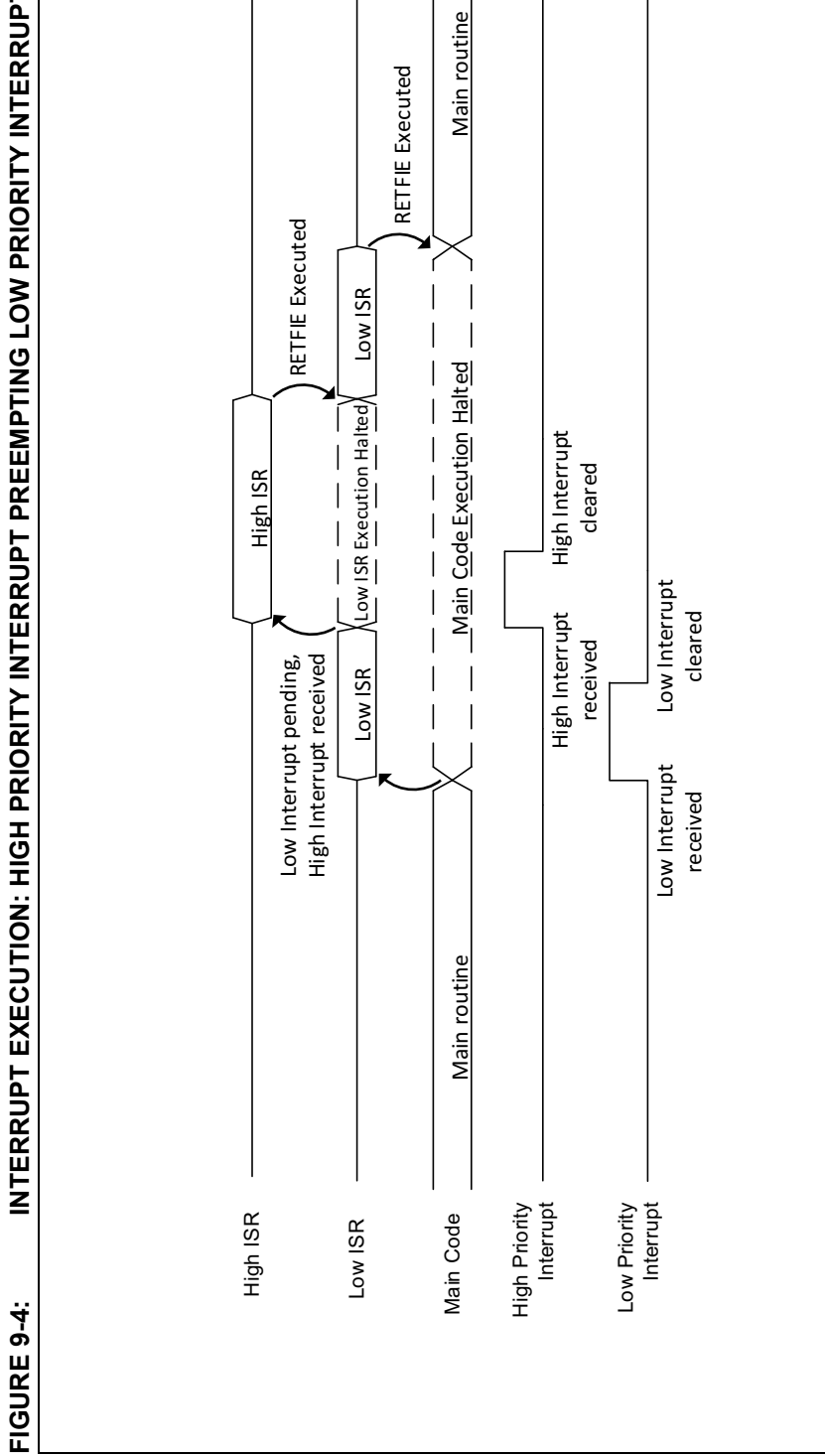


### 9.4.3 PREEMPTING LOW PRIORITY INTERRUPTS

Low-priority interrupts can be preempted by high priority interrupts. While in the low priority ISR, if a high-priority interrupt arrives, the high priority interrupt request is generated and the low priority ISR is suspended, while the high priority ISR is executed, see [Figure 9-4](#).

After the high priority ISR is complete and if any other high priority interrupt requests are not active, the execution returns to the preempted low priority ISR.

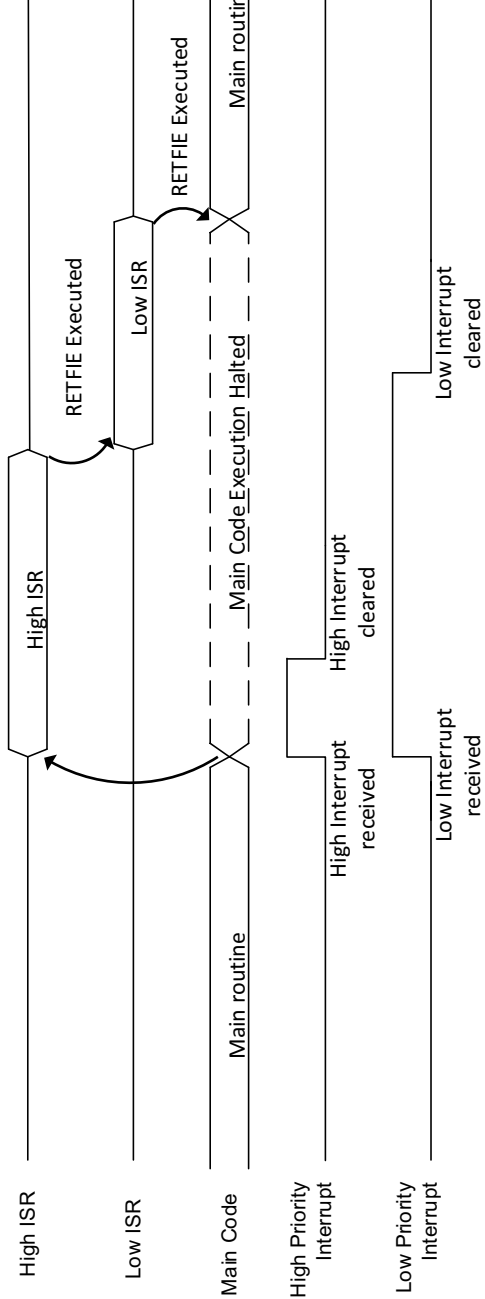
- Note 1:** The high priority interrupt flag must be cleared to avoid recursive interrupts.
- 2:** If a low-priority ISR was already servicing a high priority interrupt halfway before moving on to a high priority ISR, then the low priority ISR is completely serviced even if the user code clears GIEL.



9.4.4 SIMULTANEOUS LOW AND HIGH PRIORITY INTERRUPTS

When both high and low interrupts are active in the same instruction cycle (i.e., simultaneous interrupt events), both the high and the low priority requests are generated. The high priority ISR is serviced first before servicing the low priority interrupt see [Figure 9-5](#).

**FIGURE 9-5: INTERRUPT EXECUTION: SIMULTANEOUS LOW AND HIGH PRIORITY INTERRUPTS**



## 9.5 Context Saving

The Interrupt controller supports a two-level deep context saving (Main routine context and Low ISR context). Refer to state machine shown in [Figure 9-6](#) for details.

The Program Counter (PC) is saved on the dedicated device PC stack. CPU registers saved include STATUS, WREG, BSR, FSR0/1/2, PRODL/H and PCLATH/U.

After WREG has been saved to the context registers, the resolved vector number of the interrupt source to be serviced is copied into WREG. Context save and restore operation is completed by the interrupt controller based on current state of the interrupts and the order in which they were sent to the CPU.

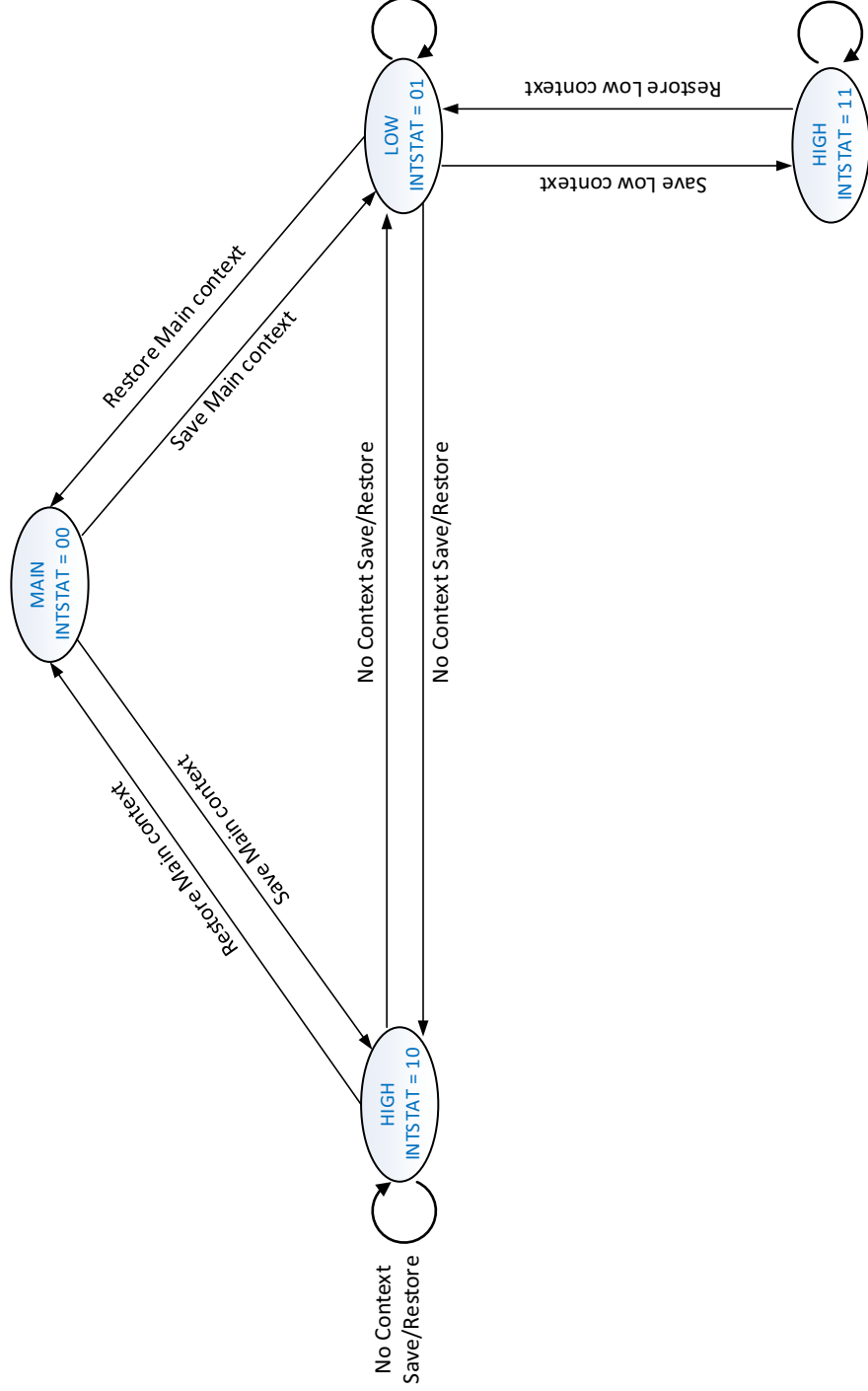
Context save/restore works the same way in both states of MVECEN. When IPEN = 0, there is only one level interrupt active. Hence, only the main context is saved when an interrupt is received.

### 9.5.1 ACCESSING SHADOW REGISTERS

The Interrupt controller automatically saves the context information in the shadow registers available in Bank 56. Both the saved context values (i.e., main routine and low ISR) can be accessed using the same set of shadow registers. By clearing the SHADLO bit in the SHADCON register ([Register 9-40](#)), the CPU register values saved for main routine context can accessed, and by setting the SHADLO bit of the CPU register, values saved for low ISR context can accessed. Low ISR context is automatically restored to the CPU registers upon exiting the high ISR. Similarly, the main context is automatically restored to the CPU registers upon exiting the low ISR.

The Shadow registers in Bank 56 are readable and writable, so if the user desires to modify the context then the corresponding shadow register should be modified and the value will be restored when exiting the ISR. Depending on the user's application, other registers may also need to be saved.

**FIGURE 9-6: CONTEXT SAVE STATE MACHINE DIAGRAM**



## 9.6 Returning from Interrupt Service Routine (ISR)

The “Return from Interrupt” instruction (`RETFIE`) is used to mark the end of an ISR.

When `RETFIE 1` instruction is executed, the PC is loaded with the saved PC value from the top of the PC stack. Saved context is also restored with the execution of this instruction. Thus, execution returns to the previous state of operation that existed before the interrupt occurred.

When `RETFIE 0` instruction is executed, the saved context is not restored back to the registers.

## 9.7 Interrupt Latency

By assigning each interrupt with a vector address/number (`MVECEN = 1`), scanning of all interrupts is not necessary to determine the source of the interrupt.

When `MVECEN = 1`, Vectored interrupt controller requires three clock cycles to vector to the ISR from main routine, thereby removing dependency of interrupt timing on compiled code.

There is a fixed latency of three instruction cycles between the completion of the instruction active when the interrupt occurred and the first instruction of the Interrupt Service Routine. [Figure 9-7](#), [Figure 9-8](#) and [Figure 9-9](#) illustrates the sequence of events when a peripheral interrupt is asserted when the last executed instruction is one-cycle, two-cycle and three-cycle respectively, when `MVECEN = 1`.

After the Interrupt Flag Status bit is set, the current instruction completes executing. In the first latency cycle, the contents of the PC, STATUS, WREG, BSR, FSR0/1/2, PRODL/H and PCLATH/U registers are context saved and the IVTBASE+ Vector number is calculated. In the second latency cycle, the PC is loaded with the calculated vector table address for the interrupt source and the starting address of the ISR is fetched. In the third latency cycle, the PC is loaded with the ISR address. All the latency cycles are executed as a `FNOP` instruction.

When `MVECEN = 0`, Vectored interrupt controller requires two clock cycles to vector to the ISR from main routine. There is a latency of two instruction cycles plus the software latency between the completion of the instruction active when the interrupt occurred and the first instruction of the Interrupt Service Routine.

FIGURE 9-7: INTERRUPT TIMING DIAGRAM – ONE CYCLE INSTRUCTION

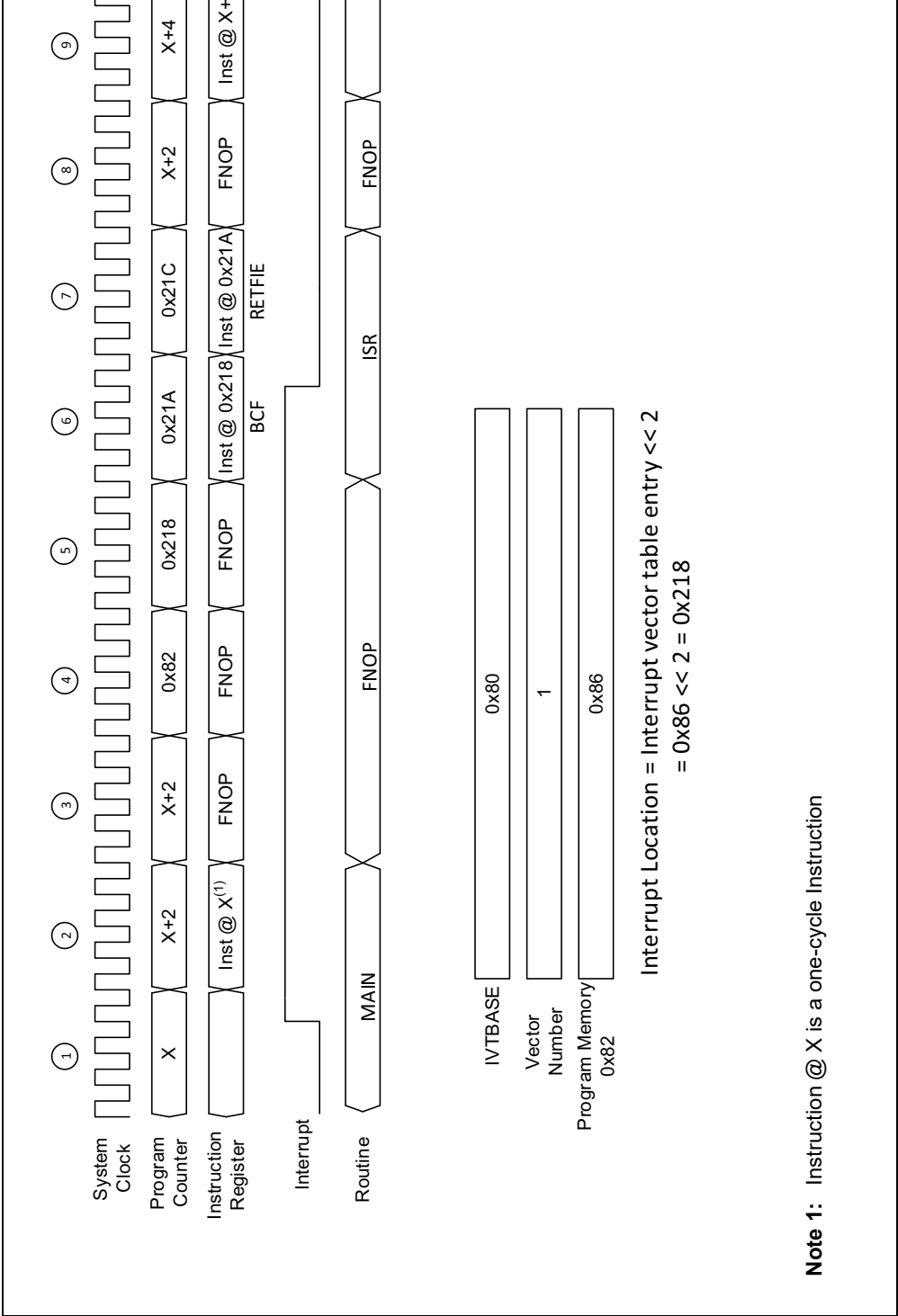
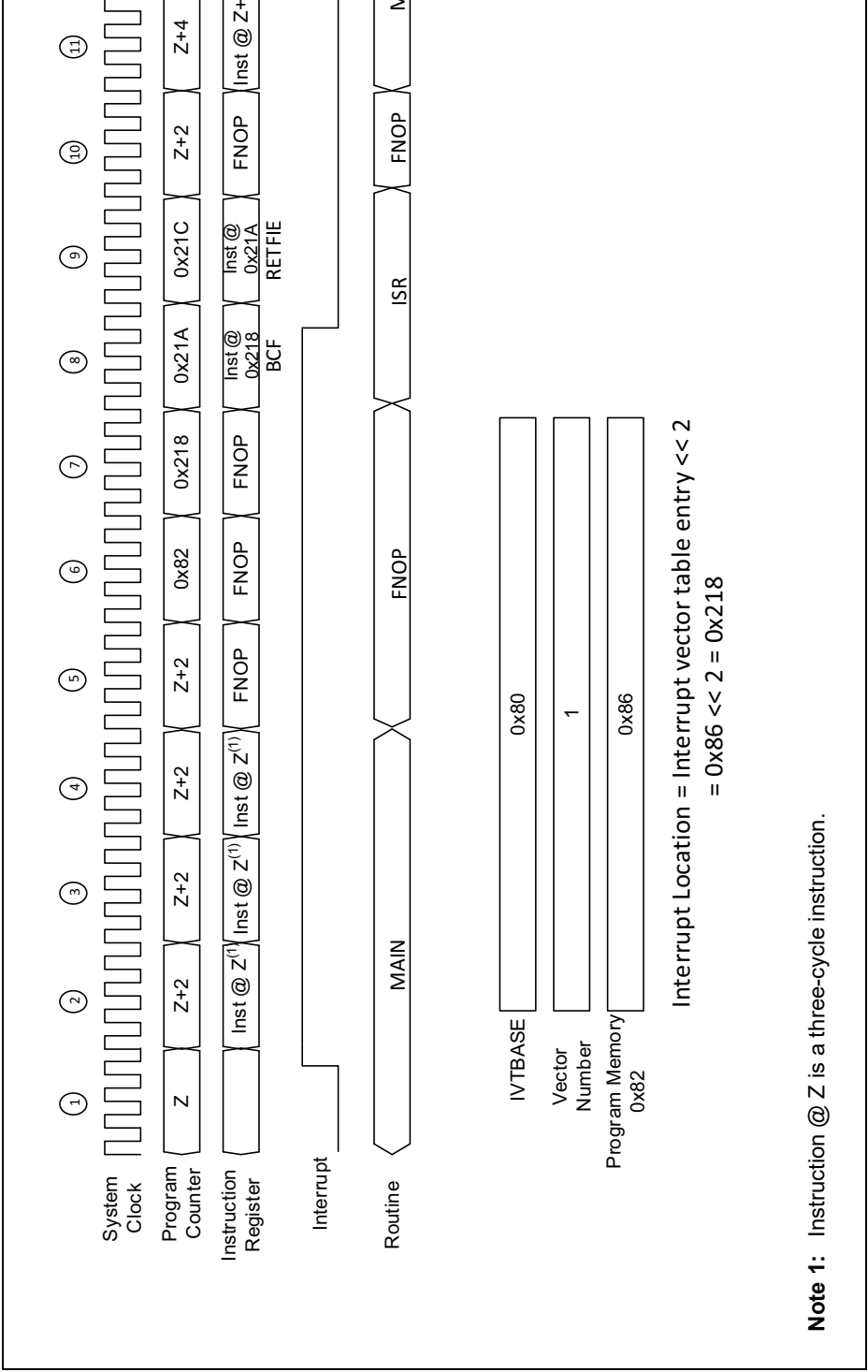




FIGURE 9-9: INTERRUPT TIMING DIAGRAM – THREE CYCLE INSTRUCTION



**Note 1:** Instruction @ Z is a three-cycle instruction.



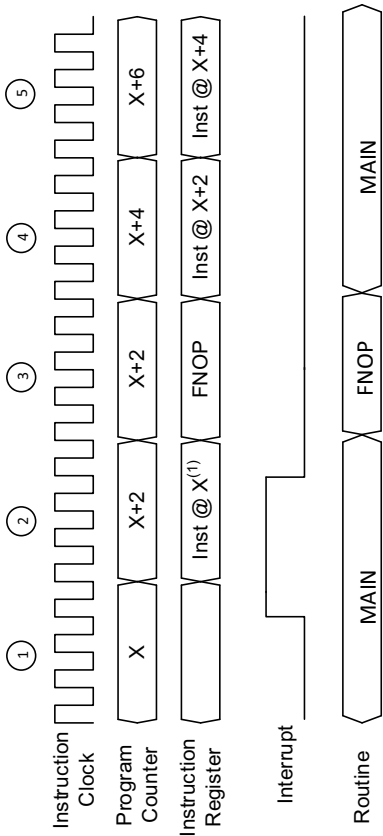
9.7.1 ABORTING INTERRUPTS

If the last instruction before the interrupt controller vectors to the ISR from main routine clears the GIE, PIE or PIR bit associated with the interrupt, the controller executes one force NOP cycle before it returns to the main routine.

Figure 9-10 illustrates the sequence of events when a peripheral interrupt is asserted and then cleared on the last executed instruction cycle.

If the GIE, PIE or PIR bit associated with the interrupt is cleared prior to vectoring to the ISR, then the controller continues executing the main routine.

FIGURE 9-10: INTERRUPT TIMING DIAGRAM - ABORTING INTERRUPTS



**Note 1:** Inst @ X clears the interrupt flag, Example BCF INTCON0, GIE.

## 9.8 Interrupt Setup Procedure

1. When using interrupt priority levels, set the IPEN bit in INTCON0 register and then select the user-assigned priority level for the interrupt source by writing the control bits in the appropriate IPRx Control register.

**Note:** At a device Reset, the IPRx registers are initialized, such that all user interrupt sources are assigned to high priority.

2. Clear the Interrupt Flag Status bit associated with the peripheral in the associated PIRx Status register.
3. Enable the interrupt source by setting the interrupt enable control bit associated with the source in the appropriate PIRx Control register.
4. If the vector table is used (MVECEN = 1), then setup the start address for the Interrupt Vector Table using the IVTBASE register. See [Section 9.2.2 “Interrupt Vector Table Contents”](#).
5. Once the IVTBASE is written to, set the Interrupt enable bits in INTCON0 register.
6. An example of setting up interrupts and ISRs using assembly and C can be found in Examples 9-3 and 9-4.

## 9.9 External Interrupt Pins

The PIC18(L)F25/26K83 devices have three external interrupt sources which can be assigned to any pin on different ports based on the PPS settings. Refer [Section 17.0 “Peripheral Pin Select \(PPS\) Module”](#) for possible rerouting options. The external interrupt sources are edge-triggered. If the corresponding INTxEDG bit in the INTCON0 register is set (= 1), the interrupt is triggered by a rising edge. If the bit is clear, the trigger is on the falling edge.

When a valid edge appears on the INTx pin, the corresponding flag bit, INTxF in the PIRx registers, is set. This interrupt can be disabled by clearing the corresponding enable bit, INTxE. Flag bit, INTxF, must be cleared by software in the Interrupt Service Routine before re-enabling the interrupt.

All external interrupts (INT0, INT1 and INT2) can wake-up the processor from Idle or Sleep modes if bit INTxE was set prior to going into those modes. If the Global Interrupt Enable bit, GIE/GIEH, is set, the processor will branch to the interrupt vector following wake-up. Interrupt priority is determined by the value contained in the interrupt priority bits, INT0IP, INT1IP and INT2IP of the IPRx registers.

## 9.10 Wake-up from Sleep

The interrupt controller provides a wake-up request to the CPU whenever an interrupt event occurs, if the interrupt event is enabled. This occurs regardless of whether the part is in Run, Idle/Doze or Sleep modes. The status of the GIEH/GIEL bits has no effect on the wake-up request. The wake-up request will be asynchronous to all clocks.

## 9.11 Interrupt Compatibility

When the MVECEN bit in Configuration Word 2L is cleared ([Register 5-3](#)), the Interrupt Vector Table feature is disabled and interrupts are compatible with previous high performance 8-bit PIC18 microcontroller devices. In this mode, the Interrupt Vector Table priority has no effect.

When the IPEN bit is also cleared, the interrupt priority feature is disabled and interrupts are compatible with PIC16 microcontroller mid-range devices. All interrupts branch to address 0008h since the interrupt priority is disabled.

## EXAMPLE 9-3: SETTING UP VECTORED INTERRUPTS USING MPASM

```
; Each ISR routine must have a predetermined origin otherwise there will be
; an assembly error because the address is not determined until link time
; which is too late to do the divide by 4 math on the address.
; Predetermined addresses must be evenly divisible by 4.

ISRCLC2 CODE      0x7E00
; CLC2 interrupt service code here.
    BANKSEL PIR7
    BCF     PIR7, CLC2IF
    RETFIE  FAST

ISRTMR0 CODE      0x7E40
; Timer0 interrupt service code here.
    BANKSEL PIR3
    BCF     PIR3, TMR0IF
    RETFIE  FAST

ISRTMR4 CODE      0x7E60
; Timer4 interrupt service code here.
    BANKSEL PIR7
    BCF     PIR7, TMR4IF
    RETFIE  FAST

IntInit:
; Disable all interrupts
BCF     INTCON0, GIE, ACCESS

; Set IVTBASE (optional - default is 0x000008)
CLRF    IVTBASEU, ACCESS
MOVLW   0x7F
MOVWF   IVTBASEH, ACCESS
CLRF    IVTBASEL, ACCESS

; Clear any interrupt flags before enabling interrupts
BANKSEL PIR7
BCF     PIR7, CLC2IF
BCF     PIR3, TMR0IF
BCF     PIR7, TMR4IF

; Enable interrupts
BANKSEL PIE7
BSF     PIE7, CLC2IE
BSF     PIE3, TMR0IE
BSF     PIE7, TMR4IE

; Set interrupt priorities if necessary
BANKSEL IPR7
BSF     INTCON0, IPEN_INTCON0, ACCESS; Enable interrupt priority
BCF     IPR7, CLC2IP                ; Make CLC2 interrupt low priority

; Enable interrupts
BSF     INTCON0, GIEH, ACCESS
BSF     INTCON0, GIEL, ACCESS

RETURN 1

; Save TMR0ISR in vector table (IVTBASE+31*2)
ISR1    CODE      0x7F3E
        DW         0x7E40>>2)                ; (TMR0ISR/4)

; Save TMR4ISR in vector table (IVTBASE+56*2)
ISR2    CODE      0x7F70
        DW         0x7E60>>2)                ; (TMR4ISR/4)

; Save CLC2ISR in vector table (IVTBASE+60*2)
ISR3    CODE      0x7F78
        DW         (0x7E00>>2)                ; (CLC2ISR/4)
```

## EXAMPLE 9-4: SETTING UP VECTORED INTERRUPTS USING XC8

```
// NOTE 1: If IVTBASE is changed from its default value of 0x000008, then the
// "base(...)" argument must be provided in the ISR. Otherwise the vector
// table will be placed at 0x0008 by default regardless of the IVTBASE value.

// NOTE 2: When MVECEN=0 and IPEN=1, a separate argument as "high_priority"
// or "low_priority" can be used to distinguish between the two ISRs.
// If the argument is not provided, the ISR is considered high priority
// by default.

// NOTE 3: Multiple interrupts can be handled by the same ISR if they are
// specified in the "irq(...)" argument. Ex: irq(IRQ_TMR0, IRQ_CCP1)

void __interrupt(irq(IRQ_TMR0), base(0x4008)) TMR0_ISR(void)
{
    PIR3bits.TMR0IF = 0;           // Clear the interrupt flag
    LATCbits.LC0 ^= 1;             // ISR code goes here
}

void __interrupt(irq(default), base(0x4008)) DEFAULT_ISR(void)
{
    // Unhandled interrupts go here
}

void INTERRUPT_Initialize (void)
{
    INTCON0bits.GIEH = 1;           // Enable high priority interrupts
    INTCON0bits.GIEL = 1;           // Enable low priority interrupts
    INTCON0bits.IPEN = 1;           // Enable interrupt priority

    PIE3bits.TMR0IE = 1;            // Enable TMR0 interrupt
    PIE4bits.TMR1IE = 1;            // Enable TMR1 interrupt

    IPR3bits.TMR0IP = 0;            // Make TMR0 interrupt low priority

    // Change IVTBASE if required
    IVTBASEU = 0x00;                // Optional
    IVTBASEH = 0x40;                // Default is 0x0008
    IVTBASEL = 0x08;
}
```

## 9.12 Register Definitions: Interrupt Control

### REGISTER 9-1: INTCON0: INTERRUPT CONTROL REGISTER 0

| R/W-0/0  | R/W-0/0 | R/W-0/0 | U-0 | U-0 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
|----------|---------|---------|-----|-----|---------|---------|---------|
| GIE/GIEH | GIEL    | IPEN    | —   | —   | INT2EDG | INT1EDG | INT0EDG |
| bit 7    |         |         |     |     |         |         | bit 0   |

#### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7 **GIE/GIEH:** Global Interrupt Enable bits

If IPEN = 0:

GIE:

1 = Enables all unmasked interrupts

0 = Disables all interrupts

If IPEN = 1:

GIEH:

1 = Enables all unmasked high priority interrupts: bit also needs to be set for enabling low priority interrupts

0 = Disables all interrupts

bit 6 **GIEL:** Global Low Priority Interrupt Enable bit

If IPEN = 0:

Reserved, read as '0'

If IPEN = 1:

GIEL:

1 = Enables all unmasked low priority interrupts, GIEH also needs to be set for low priority interrupts

0 = Disables all low priority

bit 5 **IPEN:** Interrupt Priority Enable bit

1 = Enable priority levels on interrupts

0 = Disable priority levels on interrupts; all interrupts are treated as high priority interrupts

bit 4-3 **Unimplemented:** Read as '0'

bit 2 **INT2EDG:** External Interrupt 2 Edge Select bit

1 = Interrupt on rising edge of INT2 pin

0 = Interrupt on falling edge of INT2 pin

bit 1 **INT1EDG:** External Interrupt 1 Edge Select bit

1 = Interrupt on rising edge of INT1 pin

0 = Interrupt on falling edge of INT1 pin

bit 0 **INT0EDG:** External Interrupt 0 Edge Select bit

1 = Interrupt on rising edge of INT0 pin

0 = Interrupt on falling edge of INT0 pin

## REGISTER 9-2: INTCON1: INTERRUPT CONTROL REGISTER 1

|           |       |     |     |     |     |     |       |
|-----------|-------|-----|-----|-----|-----|-----|-------|
| R-0/0     | R-0/0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0   |
| STAT<1:0> | —     | —   | —   | —   | —   | —   | —     |
| bit 7     |       |     |     |     |     |     | bit 0 |

### Legend:

HC = Bit is cleared by hardware

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

q = Value depends on condition

bit 7-6

**STAT<1:0>:** Interrupt State Status bits

11 = High priority ISR executing, high priority interrupt was received while a low priority ISR was executing

10 = High priority ISR executing, high priority interrupt was received in main routine

01 = Low priority ISR executing, low priority interrupt was received in main routine

00 = Main routine executing

bit 5-0

**Unimplemented:** Read as '0'

## REGISTER 9-3: PIR0: PERIPHERAL INTERRUPT REQUEST REGISTER 0

| R-0/0                | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0           | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 |
|----------------------|------------|------------|------------|----------------------|------------|------------|------------|
| IOCIF <sup>(2)</sup> | CRCIF      | SCANIF     | NVMIF      | CSWIF <sup>(3)</sup> | OSFIF      | HLVDIF     | SWIF       |
| bit 7                |            |            |            |                      |            |            | bit 0      |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared | HS = Bit is set in hardware                           |

|       |                                                                                                                                                                  |
|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| bit 7 | <b>IOCIF:</b> Interrupt-on-Change Interrupt Flag bit <sup>(2)</sup><br>1 = Interrupt has occurred<br>0 = Interrupt event has not occurred                        |
| bit 6 | <b>CRCIF:</b> CRC Interrupt Flag bit<br>1 = Interrupt has occurred (must be cleared by software)<br>0 = Interrupt event has not occurred                         |
| bit 5 | <b>SCANIF:</b> Memory Scanner Interrupt Flag bit<br>1 = Interrupt has occurred (must be cleared by software)<br>0 = Interrupt event has not occurred             |
| bit 4 | <b>NVMIF:</b> NVM Interrupt Flag bit<br>1 = Interrupt has occurred (must be cleared by software)<br>0 = Interrupt event has not occurred                         |
| bit 3 | <b>CSWIF:</b> Clock Switch Interrupt Flag bit <sup>(3)</sup><br>1 = Interrupt has occurred (must be cleared by software)<br>0 = Interrupt event has not occurred |
| bit 2 | <b>OSFIF:</b> Oscillator Fail Interrupt Flag bit<br>1 = Interrupt has occurred (must be cleared by software)<br>0 = Interrupt event has not occurred             |
| bit 1 | <b>HLVDIF:</b> HLVD Interrupt Flag bit<br>1 = Interrupt has occurred (must be cleared by software)<br>0 = Interrupt event has not occurred                       |
| bit 0 | <b>SWIF:</b> Software Interrupt Flag bit<br>1 = Software Interrupt Flag Enable<br>0 = Software Interrupt Flag Disable                                            |

- Note 1:** Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.
- 2:** IOCIF is a read-only bit. To clear the interrupt condition, all bits in the IOCxF registers must be cleared.
- 3:** The CSWIF interrupt will not wake the system from Sleep. The system will sleep until another interrupt causes the wake-up.

## REGISTER 9-4: PIR1: PERIPHERAL INTERRUPT REQUEST REGISTER 1

| R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0            |
|------------|------------|------------|------------|------------|------------|------------|-----------------------|
| SMT1PWAIF  | SMT1PRAIF  | SMT1IF     | C1IF       | ADTIF      | ADIF       | ZCDIF      | INT0IF <sup>(2)</sup> |
| bit 7      |            |            |            |            |            |            | bit 0                 |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared | HS = Bit is set in hardware                           |

bit 7 **SMT1PWAIF:** SMT1 Pulse-Width Acquisition Interrupt Flag bit

- 1 = Interrupt has occurred (must be cleared by software)
- 0 = Interrupt event has not occurred

bit 6 **SMT1PRAIF:** SMT1 Period Acquisition Interrupt Flag bit

- 1 = Interrupt has occurred (must be cleared by software)
- 0 = Interrupt event has not occurred

bit 5 **SMT1IF:** SMT1 Interrupt Flag bit

- 1 = Interrupt has occurred (must be cleared by software)
- 0 = Interrupt event has not occurred

bit 4 **C1IF:** CMP1 Interrupt Flag bit

- 1 = Interrupt has occurred (must be cleared by software)
- 0 = Interrupt event has not occurred

bit 3 **ADTIF:** ADC Threshold Interrupt Flag bit

- 1 = Interrupt has occurred (must be cleared by software)
- 0 = Interrupt event has not occurred

bit 2 **ADIF:** ADC Interrupt Flag bit

- 1 = Interrupt has occurred (must be cleared by software)
- 0 = Interrupt event has not occurred

bit 1 **ZCDIF:** ZCD Interrupt Flag bit

- 1 = Interrupt has occurred (must be cleared by software)
- 0 = Interrupt event has not occurred

bit 0 **INT0IF:** External Interrupt 0 Interrupt Flag bit<sup>(2)</sup>

- 1 = Interrupt has occurred (must be cleared by software)
- 0 = Interrupt event has not occurred

**Note 1:** Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

**2:** The external interrupt GPIO pin is selected by the INTxPPS register.



## REGISTER 9-5: PIR2: PERIPHERAL INTERRUPT REGISTER 2<sup>(1)</sup>

| R-0/0                   | R-0/0                 | R-0/0                   | R-0/0                   | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 |
|-------------------------|-----------------------|-------------------------|-------------------------|------------|------------|------------|------------|
| I2C1RXIF <sup>(2)</sup> | SPI1IF <sup>(3)</sup> | SPI1TXIF <sup>(4)</sup> | SPI1RXIF <sup>(4)</sup> | DMA1AIF    | DMA1ORIF   | DMA1DCNTIF | DMA1SCNTIF |
| bit 7                   |                       |                         |                         |            |            |            | bit 0      |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared | HS = Hardware set                                     |

- bit 7 **I2C1RXIF:** I<sup>2</sup>C1 Receive Interrupt Flag bit<sup>(2)</sup>  
1 = Interrupt has occurred  
0 = Interrupt event has not occurred
- bit 6 **SPI1IF:** SPI1 Interrupt Flag bit<sup>(3)</sup>  
1 = Interrupt has occurred  
0 = Interrupt event has not occurred
- bit 5 **SPI1TXIF:** SPI1 Transmit Interrupt Flag bit<sup>(4)</sup>  
1 = Interrupt has occurred  
0 = Interrupt event has not occurred
- bit 4 **SPI1RXIF:** SPI1 Receive Interrupt Flag bit<sup>(4)</sup>  
1 = Interrupt has occurred  
0 = Interrupt event has not occurred
- bit 3 **DMA1AIF:** DMA1 Abort Interrupt Flag bit  
1 = Interrupt has occurred (must be cleared by software)  
0 = Interrupt event has not occurred
- bit 2 **DMA1ORIF:** DMA1 Overrun Interrupt Flag bit  
1 = Interrupt has occurred (must be cleared by software)  
0 = Interrupt event has not occurred
- bit 1 **DMA1DCNTIF:** DMA1 Destination Count Interrupt Flag bit  
1 = Interrupt has occurred (must be cleared by software)  
0 = Interrupt event has not occurred
- bit 0 **DMA1SCNTIF:** DMA1 Source Count Interrupt Flag bit  
1 = Interrupt has occurred (must be cleared by software)  
0 = Interrupt event has not occurred

- Note 1:** Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.
- 2:** I2CxTXIF and I2CxRXIF are read-only bits. To clear the interrupt condition, the CLRBF bit in I2CxSTAT1 register must be set.
- 3:** SPIxIF is a read-only bit. To clear the interrupt condition, all bits in the SPIxINTF register must be cleared.
- 4:** SPIxTXIF and SPIxRXIF are read-only bits and cannot be set/cleared by the software.

## REGISTER 9-6: PIR3: PERIPHERAL INTERRUPT REGISTER 3<sup>(1)</sup>

| R/W/HS-0/0 | R-0/0               | R-0/0                | R-0/0                 | R-0/0                 | R-0/0                  | R-0/0                 | R-0/0                   |
|------------|---------------------|----------------------|-----------------------|-----------------------|------------------------|-----------------------|-------------------------|
| TMR0IF     | U1IF <sup>(2)</sup> | U1EIF <sup>(3)</sup> | U1TXIF <sup>(4)</sup> | U1RXIF <sup>(4)</sup> | I2C1EIF <sup>(5)</sup> | I2C1IF <sup>(6)</sup> | I2C1TXIF <sup>(7)</sup> |
| bit 7      |                     |                      |                       |                       |                        |                       | bit 0                   |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared | HS = Bit is set in hardware                           |

|       |                                                                                                                                                     |
|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| bit 7 | <b>TMR0IF:</b> TMR0 Interrupt Flag bit<br>1 = Interrupt has occurred (must be cleared by software)<br>0 = Interrupt event has not occurred          |
| bit 6 | <b>U1IF:</b> UART1 Interrupt Flag bit <sup>(2)</sup><br>1 = Interrupt has occurred<br>0 = Interrupt event has not occurred                          |
| bit 5 | <b>U1EIF:</b> UART1 Framing Error Interrupt Flag bit <sup>(3)</sup><br>1 = Interrupt has occurred<br>0 = Interrupt event has not occurred           |
| bit 4 | <b>U1TXIF:</b> UART1 Transmit Interrupt Flag bit <sup>(4)</sup><br>1 = Interrupt has occurred<br>0 = Interrupt event has not occurred               |
| bit 3 | <b>U1RXIF:</b> UART1 Receive Interrupt Flag bit <sup>(4)</sup><br>1 = Interrupt has occurred<br>0 = Interrupt event has not occurred                |
| bit 2 | <b>I2C1EIF:</b> I <sup>2</sup> C1 Error Interrupt Flag bit <sup>(5)</sup><br>1 = Interrupt has occurred<br>0 = Interrupt event has not occurred     |
| bit 1 | <b>I2C1IF:</b> I <sup>2</sup> C1 Interrupt Flag bit <sup>(6)</sup><br>1 = Interrupt has occurred<br>0 = Interrupt event has not occurred            |
| bit 0 | <b>I2C1TXIF:</b> I <sup>2</sup> C1 Transmit Interrupt Flag bit <sup>(7)</sup><br>1 = Interrupt has occurred<br>0 = Interrupt event has not occurred |

- Note 1:** Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.
- 2:** UxIF is a read-only bit. To clear the interrupt condition, all bits in the UxUIR register must be cleared.
- 3:** UxEIF is a read-only bit. To clear the interrupt condition, all bits in the UxERRIR register must be cleared.
- 4:** UxTXIF and UxRXIF are read-only bits and cannot be set/cleared by the software.
- 5:** I2CxEIF is a read-only bit. To clear the interrupt condition, all bits in the I2CxERR register must be cleared.
- 6:** I2CxIF is a read-only bit. To clear the interrupt condition, all bits in the I2CxPIR register must be cleared.
- 7:** I2CxTXIF and I2CxRXIF are read-only bits. To clear the interrupt condition, the CLRBF bit in I2CxSTAT1 register must be set.

## REGISTER 9-7: PIR4: PERIPHERAL INTERRUPT REGISTER 4<sup>(1)</sup>

| R/W/HS-0/0            | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 |
|-----------------------|------------|------------|------------|------------|------------|------------|------------|
| INT1IF <sup>(2)</sup> | CLC1IF     | CWG1IF     | NCO1IF     | CCP1IF     | TMR2IF     | TMR1GIF    | TMR1IF     |
| bit 7                 |            |            |            |            |            |            | bit 0      |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared | HS = Bit is set in hardware                           |

- bit 7      **INT1IF:** External Interrupt 1 Interrupt Flag bit<sup>(2)</sup>  
             1 = Interrupt has occurred (must be cleared by software)  
             0 = Interrupt event has not occurred
- bit 6      **CLC1IF:** CLC1 Interrupt Flag bit  
             1 = Interrupt has occurred (must be cleared by software)  
             0 = Interrupt event has not occurred
- bit 5      **CWG1IF:** CWG1 Interrupt Flag bit  
             1 = Interrupt has occurred (must be cleared by software)  
             0 = Interrupt event has not occurred
- bit 4      **NCO1IF:** NCO1 Interrupt Flag bit  
             1 = Interrupt has occurred (must be cleared by software)  
             0 = Interrupt event has not occurred
- bit 3      **CCP1IF:** CCP1 Interrupt Flag bit  
             1 = Interrupt has occurred (must be cleared by software)  
             0 = Interrupt event has not occurred
- bit 2      **TMR2IF:** TMR2 Interrupt Flag bit  
             1 = Interrupt has occurred (must be cleared by software)  
             0 = Interrupt event has not occurred
- bit 1      **TMR1GIF:** TMR1 Gate Interrupt Flag bit  
             1 = Interrupt has occurred (must be cleared by software)  
             0 = Interrupt event has not occurred
- bit 0      **TMR1IF:** TMR1 Interrupt Flag bit  
             1 = Interrupt has occurred (must be cleared by software)  
             0 = Interrupt event has not occurred

**Note 1:** Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

**2:** The external interrupt GPIO pin is selected by the INTxPPS register.

## REGISTER 9-8: PIR5: PERIPHERAL INTERRUPT REGISTER 5<sup>(1)</sup>

| R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0    | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0    | R/W/HS-0/0    |
|------------|------------|------------|---------------|------------|------------|---------------|---------------|
| IRXIF      | WAKIF      | ERRIF      | TXB2IF/TXBnIF | TXB1IF     | TXB0IF     | RXB1IF/RXBnIF | RXB0IF/FIFOIF |
| bit 7      |            |            |               |            |            |               | bit 0         |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared | HS = Bit is set in hardware                           |

- bit 7 **IRXIF**: CAN Invalid Message Received Interrupt Flag bit  
1 = Interrupt has occurred (must be cleared by software)  
0 = Interrupt event has not occurred
- bit 6 **WAKIF**: CAN Bus Wake-Up Activity Interrupt Flag bit  
1 = Interrupt has occurred (must be cleared by software)  
0 = Interrupt event has not occurred
- bit 5 **ERRIF**: CAN Error Interrupt Flag bit (Multiple sources in the COMSTAT register)  
1 = Interrupt has occurred (must be cleared by software)  
0 = Interrupt event has not occurred
- bit 4 **TXB2IF/TXBnIF**: CAN Transmit Buffer 2/Transmit Buffer n Interrupt Flag bit  
1 = Interrupt has occurred (must be cleared by software)  
0 = Interrupt event has not occurred
- bit 3 **TXB1IF**: CAN Transmit Buffer 1 Interrupt Flag bit  
1 = Interrupt has occurred (must be cleared by software)  
0 = Interrupt event has not occurred
- bit 2 **TXB0IF**: CAN Transmit Buffer 0 Interrupt Flag bit  
1 = Interrupt has occurred (must be cleared by software)  
0 = Interrupt event has not occurred
- bit 1 **RXB1IF/RXBnIF**: CAN Receive Buffer 1/ Receive Buffer n Interrupt Flag bit  
1 = Interrupt has occurred (must be cleared by software)  
0 = Interrupt event has not occurred
- bit 0 **RXB0IF/FIFOWMIF**: CAN Receive Buffer 0/FIFO Watermark Interrupt Flag bit  
1 = Interrupt has occurred (must be cleared by software)  
0 = Interrupt event has not occurred

**Note 1:** Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

## REGISTER 9-9: PIR6: PERIPHERAL INTERRUPT REGISTER 6<sup>(1)</sup>

| R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 |
|------------|------------|------------|------------|------------|------------|------------|------------|
| DMA2AIF    | DMA2ORIF   | DMA2DCNTIF | DMA2SCNTIF | SMT2PWAIF  | SMT2PRAIF  | SMT2IF     | C2IF       |
| bit 7      |            |            |            |            |            |            | bit 0      |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared | HS = Bit is set in hardware                           |

|       |                                                                                                                                                      |
|-------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| bit 7 | <b>DMA2AIF:</b> DMA2 Abort Interrupt Flag bit<br>1 = Interrupt has occurred (must be cleared by software)<br>0 = Interrupt event has not occurred    |
| bit 6 | <b>DMA2ORIF:</b> DMA2 Overrun Interrupt Flag bit<br>1 = Interrupt has occurred (must be cleared by software)<br>0 = Interrupt event has not occurred |
| bit 5 | <b>DMA2DCNTIF:</b> DMA2 Destination Count Interrupt Flag bit<br>1 = Interrupt has occurred<br>0 = Interrupt event has not occurred                   |
| bit 4 | <b>DMA2SCNTIF:</b> DMA2 Source Count Interrupt Flag bit<br>1 = Interrupt has occurred<br>0 = Interrupt event has not occurred                        |
| bit 3 | <b>SMT2PWAIF:</b> SMT2 Pulse-Width Acquisition Interrupt Flag bit<br>1 = Interrupt has occurred<br>0 = Interrupt event has not occurred              |
| bit 2 | <b>SMT2PRAIF:</b> SMT2 Period Acquisition Interrupt Flag bit<br>1 = Interrupt has occurred<br>0 = Interrupt event has not occurred                   |
| bit 1 | <b>SMT2IF:</b> SMT2 Interrupt Flag bit<br>1 = Interrupt has occurred<br>0 = Interrupt event has not occurred                                         |
| bit 0 | <b>C2IF:</b> C2 Interrupt Flag bit<br>1 = Interrupt has occurred<br>0 = Interrupt event has not occurred                                             |

**Note 1:** Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

## REGISTER 9-10: PIR7: PERIPHERAL INTERRUPT REGISTER 7<sup>(1)</sup>

| R-0/0               | R-0/0                | R-0/0                 | R-0/0                 | R-0/0                  | R-0/0                 | R-0/0                   | R-0/0                   |
|---------------------|----------------------|-----------------------|-----------------------|------------------------|-----------------------|-------------------------|-------------------------|
| U2IF <sup>(2)</sup> | U2EIF <sup>(3)</sup> | U2TXIF <sup>(4)</sup> | U2RXIF <sup>(4)</sup> | I2C2EIF <sup>(5)</sup> | I2C2IF <sup>(6)</sup> | I2C2TXIF <sup>(7)</sup> | I2C2RXIF <sup>(7)</sup> |
| bit 7               |                      |                       |                       |                        |                       |                         | bit 0                   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

HS = Bit is set in hardware

|       |                                                                                                                                                     |
|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| bit 7 | <b>U2IF:</b> UART2 Interrupt Flag bit <sup>(2)</sup><br>1 = Interrupt has occurred<br>0 = Interrupt event has not occurred                          |
| bit 6 | <b>U2EIF:</b> UART2 Framing Error Interrupt Flag bit <sup>(3)</sup><br>1 = Interrupt has occurred<br>0 = Interrupt event has not occurred           |
| bit 5 | <b>U2TXIF:</b> UART2 Transmit Interrupt Flag bit <sup>(4)</sup><br>1 = Interrupt has occurred<br>0 = Interrupt event has not occurred               |
| bit 4 | <b>U2RXIF:</b> UART2 Receive Interrupt Flag bit <sup>(4)</sup><br>1 = Interrupt has occurred<br>0 = Interrupt event has not occurred                |
| bit 3 | <b>I2C2EIF:</b> I <sup>2</sup> C2 Error Interrupt Flag bit <sup>(5)</sup><br>1 = Interrupt has occurred<br>0 = Interrupt event has not occurred     |
| bit 2 | <b>I2C2IF:</b> I <sup>2</sup> C2 Interrupt Flag bit <sup>(6)</sup><br>1 = Interrupt has occurred<br>0 = Interrupt event has not occurred            |
| bit 1 | <b>I2C2TXIF:</b> I <sup>2</sup> C2 Transmit Interrupt Flag bit <sup>(7)</sup><br>1 = Interrupt has occurred<br>0 = Interrupt event has not occurred |
| bit 0 | <b>I2C2RXIF:</b> I <sup>2</sup> C2 Receive Interrupt Flag bit <sup>(7)</sup><br>1 = Interrupt has occurred<br>0 = Interrupt event has not occurred  |

- Note 1:** Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.
- 2:** UxIF is a read-only bit. To clear the interrupt condition, all bits in the UxUIR register must be cleared.
- 3:** UxEIF is a read-only bit. To clear the interrupt condition, all bits in the UxERRIR register must be cleared.
- 4:** UxTXIF and UxRXIF are read-only bits and cannot be set/cleared by the software.
- 5:** I2CxEIF is a read-only bit. To clear the interrupt condition, all bits in the I2CxERR register must be cleared.
- 6:** I2CxIF is a read-only bit. To clear the interrupt condition, all bits in the I2CxPIR register must be cleared.
- 7:** I2CxTXIF and I2CxRXIF are read-only bits. To clear the interrupt condition, the CLRBF bit in I2CxSTAT1 register must be set.

## REGISTER 9-11: PIR8: PERIPHERAL INTERRUPT REGISTER 8<sup>(1)</sup>

| R/W/HS-0/0 | R/W/HS-0/0            | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 |
|------------|-----------------------|------------|------------|------------|------------|------------|------------|
| TMR5IF     | INT2IF <sup>(2)</sup> | CLC2IF     | CWG2IF     | CCP2IF     | TMR4IF     | TMR3GIF    | TMR3IF     |
| bit 7      |                       |            |            |            |            |            | bit 0      |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared | HS = Bit is set in hardware                           |

|       |                                                                                                                                                                           |
|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| bit 7 | <b>TMR5IF:</b> TMR5 Interrupt Flag bit<br>1 = Interrupt has occurred (must be cleared by software)<br>0 = Interrupt event has not occurred                                |
| bit 6 | <b>INT2IF:</b> External Interrupt 2 Interrupt Flag bit <sup>(2)</sup><br>1 = Interrupt has occurred (must be cleared by software)<br>0 = Interrupt event has not occurred |
| bit 5 | <b>CLC2IF:</b> CLC2 Interrupt Flag bit<br>1 = Interrupt has occurred (must be cleared by software)<br>0 = Interrupt event has not occurred                                |
| bit 4 | <b>CWG2IF:</b> CWG2 Interrupt Flag bit<br>1 = Interrupt has occurred (must be cleared by software)<br>0 = Interrupt event has not occurred                                |
| bit 3 | <b>CCP2IF:</b> CCP2 Interrupt Flag bit<br>1 = Interrupt has occurred (must be cleared by software)<br>0 = Interrupt event has not occurred                                |
| bit 2 | <b>TMR4IF:</b> TMR4 Interrupt Flag bit<br>1 = Interrupt has occurred (must be cleared by software)<br>0 = Interrupt event has not occurred                                |
| bit 1 | <b>TMR3GIF:</b> TMR3G Gate Interrupt Flag bit<br>1 = Interrupt has occurred (must be cleared by software)<br>0 = Interrupt event has not occurred                         |
| bit 0 | <b>TMR3IF:</b> TMR3 Interrupt Flag bit<br>1 = Interrupt has occurred (must be cleared by software)<br>0 = Interrupt event has not occurred                                |

**Note 1:** Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

**2:** The external interrupt GPIO pin is selected by the INTxPPS register.

## REGISTER 9-12: PIR9: PERIPHERAL INTERRUPT REGISTER 9<sup>(1)</sup>

| U-0   | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 |
|-------|------------|------------|------------|------------|------------|------------|------------|
| —     | CLC4IF     | CCP4IF     | CLC3IF     | CWG3IF     | CCP3IF     | TMR6IF     | TMR5GIF    |
| bit 7 |            |            |            |            |            |            | bit 0      |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

|       |                                                                                                                                                  |
|-------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| bit 7 | <b>Unimplemented:</b> Read as '0'                                                                                                                |
| bit 6 | <b>CLC4IF:</b> CLC4 Interrupt Flag bit<br>1 = Interrupt has occurred (must be cleared by software)<br>0 = Interrupt event has not occurred       |
| bit 5 | <b>CCP4IF:</b> CCP4 Interrupt Flag bit<br>1 = Interrupt has occurred (must be cleared by software)<br>0 = Interrupt event has not occurred       |
| bit 4 | <b>CLC3IF:</b> CLC3 Interrupt Flag bit<br>1 = Interrupt has occurred (must be cleared by software)<br>0 = Interrupt event has not occurred       |
| bit 3 | <b>CWG3IF:</b> CWG3 Interrupt Flag bit<br>1 = Interrupt has occurred (must be cleared by software)<br>0 = Interrupt event has not occurred       |
| bit 2 | <b>CCP3IF:</b> CCP3 Interrupt Flag bit<br>1 = Interrupt has occurred (must be cleared by software)<br>0 = Interrupt event has not occurred       |
| bit 1 | <b>TMR6IF:</b> TMR6 Interrupt Flag bit<br>1 = Interrupt has occurred (must be cleared by software)<br>0 = Interrupt event has not occurred       |
| bit 0 | <b>TMR5GIF:</b> TMR5 Gate Interrupt Flag bit<br>1 = Interrupt has occurred (must be cleared by software)<br>0 = Interrupt event has not occurred |

**Note 1:** Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.



## REGISTER 9-13: PIE0: PERIPHERAL INTERRUPT ENABLE REGISTER 0

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| IOIE    | CRCIE   | SCANIE  | NVMIE   | CSWIE   | OSFIE   | HLVDIE  | SWIE    |
| bit 7   |         |         |         |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

|       |                                                                                   |
|-------|-----------------------------------------------------------------------------------|
| bit 7 | <b>IOIE:</b> Interrupt-on-Change Enable bit<br>1 = Enabled<br>0 = Disabled        |
| bit 6 | <b>CRCIE:</b> CRC Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled             |
| bit 5 | <b>SCANIE:</b> Memory Scanner Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled |
| bit 4 | <b>NVMIE:</b> NVM Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled             |
| bit 3 | <b>CSWIE:</b> Clock Switch Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled    |
| bit 2 | <b>OSFIE:</b> Oscillator Fail Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled |
| bit 1 | <b>HLVDIE:</b> HLVD Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled           |
| bit 0 | <b>SWIE:</b> Software Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled         |

## REGISTER 9-14: PIE1: PERIPHERAL INTERRUPT ENABLE REGISTER 1

| R/W-0/0   | R/W-0/0   | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|-----------|-----------|---------|---------|---------|---------|---------|---------|
| SMT1PWAIE | SMT1PRAIE | SMT1IE  | C1IE    | ADTIE   | ADIE    | ZCDIE   | INT0IE  |
| bit 7     |           |         |         |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7 **SMT1PWAIE:** SMT1 Pulse Width Acquisition Interrupt Enable bit

1 = Enabled

0 = Disabled

bit 6 **SMT1PRAIE:** SMT1 Period Acquisition Interrupt Enable bit

1 = Enabled

0 = Disabled

bit 5 **SMT1IE:** SMT1 Interrupt Enable bit

1 = Enabled

0 = Disabled

bit 4 **C1IE:** C1 Interrupt Enable bit

1 = Enabled

0 = Disabled

bit 3 **ADTIE:** ADC Threshold Interrupt Enable bit

1 = Enabled

0 = Disabled

bit 2 **ADIE:** ADC Interrupt Enable bit

1 = Enabled

0 = Disabled

bit 1 **ZCDIE:** ZCD Interrupt Enable bit

1 = Enabled

0 = Disabled

bit 0 **INT0IE:** External Interrupt 0 Enable bit

1 = Enabled

0 = Disabled

## REGISTER 9-15: PIE2: PERIPHERAL INTERRUPT ENABLE REGISTER 2

|          |         |          |          |         |          |            |            |
|----------|---------|----------|----------|---------|----------|------------|------------|
| R/W-0/0  | R/W-0/0 | R/W-0/0  | R/W-0/0  | R/W-0/0 | R/W-0/0  | R/W-0/0    | R/W-0/0    |
| I2C1RXIE | SPI1IE  | SPI1TXIE | SPI1RXIE | DMA1AIE | DMA1ORIE | DMA1DCNTIE | DMA1SCNTIE |
| bit 7    |         |          |          |         |          |            | bit 0      |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

- bit 7      **I2C1RXIE:** I<sup>2</sup>C1 Receive Interrupt Enable bit  
1 = Enabled  
0 = Disabled
- bit 6      **SPI1IE:** SPI1 Interrupt Enable bit  
1 = Enabled  
0 = Disabled
- bit 5      **SPI1TXIE:** SPI1 Transmit Interrupt Enable bit  
1 = Enabled  
0 = Disabled
- bit 4      **SPI1RXIE:** SPI1 Receive Interrupt Enable bit  
1 = Enabled  
0 = Disabled
- bit 3      **DMA1AIE:** DMA1 Abort Interrupt Enable bit  
1 = Enabled  
0 = Disabled
- bit 2      **DMA1ORIE:** DMA1 Overrun Interrupt Enable bit  
1 = Enabled  
0 = Disabled
- bit 1      **DMA1DCNTIE:** DMA1 Destination Count Interrupt Enable bit  
1 = Enabled  
0 = Disabled
- bit 0      **DMA1SCNTIE:** DMA1 Source Count Interrupt Enable bit  
1 = Enabled  
0 = Disabled

**REGISTER 9-16: PIE3: PERIPHERAL INTERRUPT ENABLE REGISTER 3**

|         |         |         |         |         |         |         |          |
|---------|---------|---------|---------|---------|---------|---------|----------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0  |
| TMR0IE  | U1IE    | U1EIE   | U1TXIE  | U1RXIE  | I2C1EIE | I2C1IE  | I2C1TXIE |
| bit 7   |         |         |         |         |         |         | bit 0    |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

|       |                                                                                                 |
|-------|-------------------------------------------------------------------------------------------------|
| bit 7 | <b>TMR0IE:</b> TMR0 Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled                         |
| bit 6 | <b>U1IE:</b> UART1 Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled                          |
| bit 5 | <b>U1EIE:</b> UART1 Framing Error Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled           |
| bit 4 | <b>U1TXIE:</b> UART1 Transmit Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled               |
| bit 3 | <b>U1RXIE:</b> UART1 Receive Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled                |
| bit 2 | <b>I2C1EIE:</b> I <sup>2</sup> C1 Error Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled     |
| bit 1 | <b>I2C1IE:</b> I <sup>2</sup> C1 Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled            |
| bit 0 | <b>I2C1TXIE:</b> I <sup>2</sup> C1 Transmit Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled |

## REGISTER 9-17: PIE4: PERIPHERAL INTERRUPT ENABLE REGISTER 4

|         |         |         |         |         |         |         |         |
|---------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| INT1IE  | CLC1IE  | CWG1IE  | NCO1IE  | CCP1IE  | TMR2IE  | TMR1GIE | TMR1IE  |
| bit 7   |         |         |         |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7 **INT1IE:** External Interrupt 1 Interrupt Enable bit

1 = Enabled

0 = Disabled

bit 6 **CLC1IE:** CLC1 Interrupt Enable bit

1 = Enabled

0 = Disabled

bit 5 **CWG1IE:** CWG1 Interrupt Enable bit

1 = Enabled

0 = Disabled

bit 4 **NCO1IE:** NCO1 Interrupt Enable bit

1 = Enabled

0 = Disabled

bit 3 **CCP1IE:** CCP1 Interrupt Enable bit

1 = Enabled

0 = Disabled

bit 2 **TMR2IE:** TMR2 Interrupt Enable bit

1 = Enabled

0 = Disabled

bit 1 **TMR1GIE:** TMR1 Gate Interrupt Enable bit

1 = Enabled

0 = Disabled

bit 0 **TMR1IE:** TMR1 Interrupt Enable bit

1 = Enabled

0 = Disabled

## REGISTER 9-18: PIE5: PERIPHERAL INTERRUPT ENABLE REGISTER 5

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0       | R/W-0/0 | R/W-0/0 | R/W-0/0       | R/W-0/0       |
|---------|---------|---------|---------------|---------|---------|---------------|---------------|
| IRXIE   | WAKIE   | ERRIE   | TXB2IE/TXBnIE | TXB1IE  | TXB0IE  | RXB1IE/RXBnIE | RXB0IE/FIFOIE |
| bit 7   |         |         |               |         |         |               | bit 0         |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

|       |                                                                                                                                 |
|-------|---------------------------------------------------------------------------------------------------------------------------------|
| bit 7 | <b>IRXIE:</b> CAN Invalid Message Received Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled                                  |
| bit 6 | <b>WAKIE:</b> CAN Bus Wake-up Activity Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled                                      |
| bit 5 | <b>ERRIE:</b> CAN Error Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled                                                     |
| bit 4 | <b>TXB2IE/TXBnIE:</b> CAN Transmit Buffer 2/Transmit Buffer n Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled               |
| bit 3 | <b>TXB1IE:</b> CAN Transmit Buffer 1 Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled                                        |
| bit 2 | <b>TXB0IE:</b> CAN Transmit Buffer 0 Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled                                        |
| bit 1 | <b>RXB1IE/RXBnIE:</b> CAN Receive Buffer 1/ Receive Buffer n Interrupt Flag Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled |
| bit 0 | <b>RXB0IE/FIFOIE:</b> CAN Receive Buffer 0/FIFO Full Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled                        |

## REGISTER 9-19: PIE6: PERIPHERAL INTERRUPT ENABLE REGISTER 6

|         |          |            |            |           |           |         |         |
|---------|----------|------------|------------|-----------|-----------|---------|---------|
| R/W-0/0 | R/W-0/0  | R/W-0/0    | R/W-0/0    | R/W-0/0   | R/W-0/0   | R/W-0/0 | R/W-0/0 |
| DMA2AIE | DMA2ORIE | DMA2DCNTIE | DMA2SCNTIE | SMT2PWAIE | SMT2PRAIE | SMT2IE  | C2IE    |
| bit 7   |          |            |            |           |           |         | bit 0   |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

|       |                                                                                                       |
|-------|-------------------------------------------------------------------------------------------------------|
| bit 7 | <b>DMA2AIE:</b> DMA Abort Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled                         |
| bit 6 | <b>DMA2ORIE:</b> DMA2 Overrun Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled                     |
| bit 5 | <b>DMA2DCNTIE:</b> DMA2 Destination Count Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled         |
| bit 4 | <b>DMA2SCNTIE:</b> DMA2 Source Count Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled              |
| bit 3 | <b>SMT2PWAIE:</b> SMT2 Pulse-Width Acquisition Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled    |
| bit 2 | <b>SMT2PRAIE:</b> SMT2 Period Acquisition Receive Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled |
| bit 1 | <b>SMT2IE:</b> SMT2 Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled                               |
| bit 0 | <b>C2IE:</b> C2 Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled                                   |

## REGISTER 9-20: PIE7: PERIPHERAL INTERRUPT ENABLE REGISTER 7

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0  | R/W-0/0  |
|---------|---------|---------|---------|---------|---------|----------|----------|
| U2IE    | U2EIE   | U2TXIE  | U2RXIE  | I2C2EIE | I2C2IE  | I2C2TXIE | I2C2RXIE |
| bit 7   |         |         |         |         |         |          | bit 0    |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

|       |                                                                                                 |
|-------|-------------------------------------------------------------------------------------------------|
| bit 7 | <b>U2IE:</b> UART2 Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled                          |
| bit 6 | <b>U2EIE:</b> UART2 Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled                         |
| bit 5 | <b>U2TXIE:</b> UART2 Transmit Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled               |
| bit 4 | <b>U2RXIE:</b> UART2 Receive Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled                |
| bit 3 | <b>I2C2EIE:</b> I <sup>2</sup> C2 Error Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled     |
| bit 2 | <b>I2C2IE:</b> I <sup>2</sup> C2 Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled            |
| bit 1 | <b>I2C2TXIE:</b> I <sup>2</sup> C2 Transmit Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled |
| bit 0 | <b>I2C2RXIE:</b> I <sup>2</sup> C2 Receive Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled  |



## REGISTER 9-21: PIE8: PERIPHERAL INTERRUPT ENABLE REGISTER 8

|         |         |         |         |         |         |         |         |
|---------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| TMR5IE  | INT2IE  | CLC2IE  | CWG2IE  | CCP2IE  | TMR4IE  | TMR3GIE | TMR3IE  |
| bit 7   |         |         |         |         |         | bit 0   |         |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

|       |                                                                                         |
|-------|-----------------------------------------------------------------------------------------|
| bit 7 | <b>TMR5IE:</b> TMR5 Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled                 |
| bit 6 | <b>INT2IE:</b> External Interrupt 2 Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled |
| bit 5 | <b>CLC2IE:</b> CLC2 Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled                 |
| bit 4 | <b>CWG2IE:</b> CWG2 Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled                 |
| bit 3 | <b>CCP2IE:</b> CCP2 Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled                 |
| bit 2 | <b>TMR4IE:</b> TMR4 Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled                 |
| bit 1 | <b>TMR3GIE:</b> TMR3 Gate Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled           |
| bit 0 | <b>TMR3IE:</b> TMR3 Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled                 |

## REGISTER 9-22: PIE9: PERIPHERAL INTERRUPT ENABLE REGISTER 9

| U-0   | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|-------|---------|---------|---------|---------|---------|---------|---------|
| —     | CLC4IE  | CCP4IE  | CLC3IE  | CWG3IE  | CCP3IE  | TMR6IE  | TMR5GIE |
| bit 7 |         |         |         |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

|       |                                                                          |
|-------|--------------------------------------------------------------------------|
| bit 7 | <b>Unimplemented:</b> Read as '0'                                        |
| bit 6 | <b>CLC4IE:</b> CLC4 Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled  |
| bit 5 | <b>CCP4IE:</b> CCP4 Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled  |
| bit 4 | <b>CLC3IE:</b> CLC3 Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled  |
| bit 3 | <b>CWG3IE:</b> CWG3 Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled  |
| bit 2 | <b>CCP3IE:</b> CCP3 Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled  |
| bit 1 | <b>TMR6IE:</b> TMR6 Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled  |
| bit 0 | <b>TMR5GIE:</b> TMR5 Interrupt Enable bit<br>1 = Enabled<br>0 = Disabled |

## REGISTER 9-23: IPR0: PERIPHERAL INTERRUPT PRIORITY REGISTER 0

| R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| IOCIP   | CRCIP   | SCANIP  | NVMIP   | CSWIP   | OSFIP   | HLVDIP  | SWIP    |
| bit 7   |         |         |         |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

|       |                                                                                               |
|-------|-----------------------------------------------------------------------------------------------|
| bit 7 | <b>IOCIP:</b> Interrupt-on-Change Priority bit<br>1 = High priority<br>0 = Low priority       |
| bit 6 | <b>CRCIP:</b> CRC Interrupt Priority bit<br>1 = High priority<br>0 = Low priority             |
| bit 5 | <b>SCANIP:</b> Memory Scanner Interrupt Priority bit<br>1 = High priority<br>0 = Low priority |
| bit 4 | <b>NVMIP:</b> NVM Interrupt Priority bit<br>1 = High priority<br>0 = Low priority             |
| bit 3 | <b>CSWIP:</b> Clock Switch Interrupt Priority bit<br>1 = High priority<br>0 = Low priority    |
| bit 2 | <b>OSFIP:</b> Oscillator Fail Interrupt Priority bit<br>1 = High priority<br>0 = Low priority |
| bit 1 | <b>HLVDIP:</b> HLVD Interrupt Priority bit<br>1 = High priority<br>0 = Low priority           |
| bit 0 | <b>SWIP:</b> Software Interrupt Priority bit<br>1 = High priority<br>0 = Low priority         |

**REGISTER 9-24: IPR1: PERIPHERAL INTERRUPT PRIORITY REGISTER 1**

| R/W-1/1   | R/W-1/1   | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
|-----------|-----------|---------|---------|---------|---------|---------|---------|
| SMT1PWAIP | SMT1PRAIP | SMT1IP  | C1IP    | ADTIP   | ADIP    | ZCDIP   | INT0IP  |
| bit 7     |           |         |         |         |         |         | bit 0   |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7 **SMT1PWAIP:** SMT1 Pulse Width Acquisition Interrupt Priority bit

1 = High priority

0 = Low priority

bit 6 **SMT1PRAIP:** SMT1 Period Acquisition Interrupt Priority bit

1 = High priority

0 = Low priority

bit 5 **SMT1IP:** SMT1 Interrupt Priority bit

1 = High priority

0 = Low priority

bit 4 **C1IP:** C1 Interrupt Priority bit

1 = High priority

0 = Low priority

bit 3 **ADTIP:** ADC Threshold Interrupt Priority bit

1 = High priority

0 = Low priority

bit 2 **ADIP:** ADC Interrupt Priority bit

1 = High priority

0 = Low priority

bit 1 **ZCDIP:** ZCD Interrupt Priority bit

1 = High priority

0 = Low priority

bit 0 **INT0IP:** External Interrupt 0 Interrupt Priority bit

1 = High priority

0 = Low priority

**REGISTER 9-25: IPR2: PERIPHERAL INTERRUPT PRIORITY REGISTER 2**

| R/W-1/1  | R/W-1/1 | R/W-1/1  | R/W-1/1  | R/W-1/1 | R/W-1/1  | R/W-1/1    | R/W-1/1    |
|----------|---------|----------|----------|---------|----------|------------|------------|
| I2C1RXIP | SPI1IP  | SPI1TXIP | SPI1RXIP | DMA1AIP | DMA1ORIP | DMA1DCNTIP | DMA1SCNTIP |
| bit 7    |         |          |          |         |          |            | bit 0      |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

- bit 7      **I2C1RXIP:** I<sup>2</sup>C1 Receive Interrupt Priority bit  
             1 = High priority  
             0 = Low priority
- bit 6      **SPI1IP:** SPI1 Transmit Interrupt Priority bit  
             1 = High priority  
             0 = Low priority
- bit 5      **SPI1TXIP:** I<sup>2</sup>C1 Transmit Interrupt Priority bit  
             1 = High priority  
             0 = Low priority
- bit 4      **SPI1RXIP:** SPI1 Receive Interrupt Priority bit  
             1 = High priority  
             0 = Low priority
- bit 3      **DMA1AIP:** DMA1 Abort Transmit Interrupt Priority bit  
             1 = High priority  
             0 = Low priority
- bit 2      **DMA1ORIP:** DMA1 Overrun Interrupt Priority bit  
             1 = High priority  
             0 = Low priority
- bit 1      **DMA1DCNTIP:** DMA1 Destination Count Interrupt Priority bit  
             1 = High priority  
             0 = Low priority
- bit 0      **DMA1SCNTIP:** DMA1 Source Count Interrupt Priority bit  
             1 = High priority  
             0 = Low priority

**REGISTER 9-26: IPR3: PERIPHERAL INTERRUPT PRIORITY REGISTER 3**

| R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1  |
|---------|---------|---------|---------|---------|---------|---------|----------|
| TMR0IP  | U1IP    | U1EIP   | U1TXIP  | U1RXIP  | I2C1EIP | I2C1IP  | I2C1TXIP |
| bit 7   |         |         |         |         |         |         | bit 0    |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

|       |                                                                                                             |
|-------|-------------------------------------------------------------------------------------------------------------|
| bit 7 | <b>TMR0IP:</b> TMR0 Interrupt Priority bit<br>1 = High priority<br>0 = Low priority                         |
| bit 6 | <b>U1IP:</b> UART1 Interrupt Priority bit<br>1 = High priority<br>0 = Low priority                          |
| bit 5 | <b>U1EIP:</b> UART1 Framing Error Interrupt Priority bit<br>1 = High priority<br>0 = Low priority           |
| bit 4 | <b>U1TXIP:</b> UART1 Transmit Interrupt Priority bit<br>1 = High priority<br>0 = Low priority               |
| bit 3 | <b>U1RXIP:</b> UART1 Receive Interrupt Priority bit<br>1 = High priority<br>0 = Low priority                |
| bit 2 | <b>I2C1EIP:</b> I <sup>2</sup> C1 Error Interrupt Priority bit<br>1 = High priority<br>0 = Low priority     |
| bit 1 | <b>I2C1IP:</b> I <sup>2</sup> C1 Interrupt Priority bit<br>1 = High priority<br>0 = Low priority            |
| bit 0 | <b>I2C1TXIP:</b> I <sup>2</sup> C1 Transmit Interrupt Priority bit<br>1 = High priority<br>0 = Low priority |

## REGISTER 9-27: IPR4: PERIPHERAL INTERRUPT PRIORITY REGISTER 4

| R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| INT1IP  | CLC1IP  | CWG1IP  | NCO1IP  | CCP1IP  | TMR2IP  | TMR1GIP | TMR1IP  |
| bit 7   |         |         |         |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7 **INT1IP:** External Interrupt 1 Interrupt Priority bit

1 = High priority

0 = Low priority

bit 6 **CLC1IP:** CLC1 Interrupt Priority bit

1 = High priority

0 = Low priority

bit 5 **CWG1IP:** CWG1 Interrupt Priority bit

1 = High priority

0 = Low priority

bit 4 **NCO1IP:** NCO1 Interrupt Priority bit

1 = High priority

0 = Low priority

bit 3 **CCP1IP:** CCP1 Interrupt Priority bit

1 = High priority

0 = Low priority

bit 2 **TMR2IP:** TMR2 Interrupt Priority bit

1 = High priority

0 = Low priority

bit 1 **TMR1GIP:** TMR1 Gate Interrupt Priority bit

1 = High priority

0 = Low priority

bit 0 **TMR1IP:** TMR1 Interrupt Priority bit

1 = High priority

0 = Low priority

## REGISTER 9-28: IPR5: PERIPHERAL INTERRUPT PRIORITY REGISTER 5

| R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1       | R/W-1/1 | R/W-1/1 | R/W-1/1       | R/W-1/1       |
|---------|---------|---------|---------------|---------|---------|---------------|---------------|
| IRXIP   | WAKIP   | ERRIP   | TXB2IP/TXBnIP | TXB1IP  | TXB0IP  | RXB1IP/RXBnIP | RXB0IP/FIFOIP |
| bit 7   |         |         |               |         |         |               | bit 0         |

### Legend:

R = Readable bit      W = Writable bit      U = Unimplemented bit, read as '0'  
 u = Bit is unchanged      x = Bit is unknown      -n/n = Value at POR and BOR/Value at all other Resets  
 '1' = Bit is set      '0' = Bit is cleared

- bit 7      **IRXIP:** CAN Invalid Message Received Interrupt Priority bit  
           1 = High priority  
           0 = Low priority
- bit 6      **WAKIP:** CAN Bus Wake-Up Activity Interrupt Priority bit  
           1 = High priority  
           0 = Low priority
- bit 5      **ERRIP:** CAN Error Interrupt Priority bit (Multiple Sources in the COMSTAT Register)  
           1 = High priority  
           0 = Low priority
- bit 4      **TXB2IP/TXBnIP:** CAN Transmit Buffer 2/Transmit Buffer n Interrupt Priority bit  
           1 = High priority  
           0 = Low priority
- bit 3      **TXB1IP:** CAN Transmit Buffer 1 Interrupt Priority bit  
           1 = High priority  
           0 = Low priority
- bit 2      **TXB0IP:** CAN Transmit Buffer 0 Interrupt Priority bit  
           1 = High priority  
           0 = Low priority
- bit 1      **RXB1IP/RXBnIP:** CAN Receive Buffer 1/Receive Buffer n Interrupt Priority bit  
           1 = High priority  
           0 = Low priority
- bit 0      **RXB0IP/FIFOIP:** CAN Receive Buffer 0/FIFO Full Interrupt Priority bit  
           1 = High priority  
           0 = Low priority



## REGISTER 9-29: IPR6: PERIPHERAL INTERRUPT PRIORITY REGISTER 6

|         |          |            |            |           |           |         |         |
|---------|----------|------------|------------|-----------|-----------|---------|---------|
| R/W-1/1 | R/W-1/1  | R/W-1/1    | R/W-1/1    | R/W-1/1   | R/W-1/1   | R/W-1/1 | R/W-1/1 |
| DMA2AIP | DMA2ORIP | DMA2DCNTIP | DMA2SCNTIP | SMT2PWAIP | SMT2PRAIP | SMT2IP  | C2IP    |
| bit 7   |          |            |            |           |           |         | bit 0   |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

|       |                                                                                                                |
|-------|----------------------------------------------------------------------------------------------------------------|
| bit 7 | <b>DMA2AIP:</b> DMA2 Abort Interrupt Priority bit<br>1 = High priority<br>0 = Low priority                     |
| bit 6 | <b>DMA2ORIP:</b> DMA2 Overrun Interrupt Priority bit<br>1 = High priority<br>0 = Low priority                  |
| bit 5 | <b>DMA2DCNTIP:</b> DMA2 Destination Count Interrupt Priority bit<br>1 = High priority<br>0 = Low priority      |
| bit 4 | <b>DMA2SCNTIP:</b> DMA2 Source Count Interrupt Priority bit<br>1 = High priority<br>0 = Low priority           |
| bit 3 | <b>SMT2PWAIP:</b> SMT2 Pulse-Width Acquisition Interrupt Priority bit<br>1 = High priority<br>0 = Low priority |
| bit 2 | <b>SMT2PRAIP:</b> SMT2 Period Acquisition Interrupt Priority bit<br>1 = High priority<br>0 = Low priority      |
| bit 1 | <b>SMT2IP:</b> SMT2 Interrupt Priority bit<br>1 = High priority<br>0 = Low priority                            |
| bit 0 | <b>C2IP:</b> C2 Interrupt Priority bit<br>1 = High priority<br>0 = Low priority                                |

## REGISTER 9-30: IPR7: PERIPHERAL INTERRUPT PRIORITY REGISTER 7

| R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1  | R/W-1/1  |
|---------|---------|---------|---------|---------|---------|----------|----------|
| U2IP    | U2EIP   | U2TXIP  | U2RXIP  | I2C2EIP | I2C2IP  | I2C2TXIP | I2C2RXIP |
| bit 7   |         |         |         |         |         |          | bit 0    |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

|       |                                                                                                             |
|-------|-------------------------------------------------------------------------------------------------------------|
| bit 7 | <b>U2IP:</b> UART2 Interrupt Priority bit<br>1 = High priority<br>0 = Low priority                          |
| bit 6 | <b>U2EIP:</b> UART2 Framing Error Interrupt Priority bit<br>1 = High priority<br>0 = Low priority           |
| bit 5 | <b>U2TXIP:</b> UART2 Transmit Interrupt Priority bit<br>1 = High priority<br>0 = Low priority               |
| bit 4 | <b>U2RXIP:</b> UART2 Receive Interrupt Priority bit<br>1 = High priority<br>0 = Low priority                |
| bit 3 | <b>I2C2EIP:</b> I <sup>2</sup> C2 Error Interrupt Priority bit<br>1 = High priority<br>0 = Low priority     |
| bit 2 | <b>I2C2IP:</b> I <sup>2</sup> C2 Interrupt Priority bit<br>1 = High priority<br>0 = Low priority            |
| bit 1 | <b>I2C2TXIP:</b> I <sup>2</sup> C2 Transmit Interrupt Priority bit<br>1 = High priority<br>0 = Low priority |
| bit 0 | <b>I2C2RXIP:</b> TMR4 Interrupt Priority bit<br>1 = High priority<br>0 = Low priority                       |

## REGISTER 9-31: IPR8: PERIPHERAL INTERRUPT PRIORITY REGISTER 8

| R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| TMR5IP  | INT2IP  | CLC2IP  | CWG2IP  | CCP2IP  | TMR4IP  | TMR3GIP | TMR3IP  |
| bit 7   |         |         |         |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

|       |                                                                                                     |
|-------|-----------------------------------------------------------------------------------------------------|
| bit 7 | <b>TMR5IP:</b> TMR5 Interrupt Priority bit<br>1 = High priority<br>0 = Low priority                 |
| bit 6 | <b>INT2IP:</b> External Interrupt 2 Interrupt Priority bit<br>1 = High priority<br>0 = Low priority |
| bit 5 | <b>CLC2IP:</b> CLC2 Interrupt Priority bit<br>1 = High priority<br>0 = Low priority                 |
| bit 4 | <b>CWG2IP:</b> CWG2 Interrupt Priority bit<br>1 = High priority<br>0 = Low priority                 |
| bit 3 | <b>CCP2IP:</b> CCP2 Interrupt Priority bit<br>1 = High priority<br>0 = Low priority                 |
| bit 2 | <b>TMR4IP:</b> TMR4 Interrupt Priority bit<br>1 = High priority<br>0 = Low priority                 |
| bit 1 | <b>TMR3GIP:</b> TMR3 Gate Interrupt Priority bit<br>1 = High priority<br>0 = Low priority           |
| bit 0 | <b>TMR3IP:</b> TMR3 Interrupt Priority bit<br>1 = High priority<br>0 = Low priority                 |

## REGISTER 9-32: IPR9: PERIPHERAL INTERRUPT PRIORITY REGISTER 9

| U-0   | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
|-------|---------|---------|---------|---------|---------|---------|---------|
| —     | CLC4IP  | CCP4IP  | CLC3IP  | CWG3IP  | CCP3IP  | TMR6IP  | TMR5GIP |
| bit 7 |         |         |         |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

|       |                                                                                       |
|-------|---------------------------------------------------------------------------------------|
| bit 7 | <b>Unimplemented:</b> Read as '0'                                                     |
| bit 6 | <b>CLC4IP:</b> CLC4 Interrupt Priority bit<br>1 = High priority<br>0 = Low priority   |
| bit 5 | <b>CCP4IP:</b> CCP4 Interrupt Priority bit<br>1 = High priority<br>0 = Low priority   |
| bit 4 | <b>CLC3IP:</b> CLC3 Interrupt Priority bit<br>1 = High priority<br>0 = Low priority   |
| bit 3 | <b>CWG3IP:</b> CWG3 Interrupt Priority bit<br>1 = High priority<br>0 = Low priority   |
| bit 2 | <b>CCP3IP:</b> CCP3 Interrupt Priority bit<br>1 = High priority<br>0 = Low priority   |
| bit 1 | <b>TMR6IP:</b> TMR6IP Interrupt Priority bit<br>1 = High priority<br>0 = Low priority |
| bit 0 | <b>TMR5GIP:</b> TMR5 Interrupt Priority bit<br>1 = High priority<br>0 = Low priority  |

## REGISTER 9-33: IVTBASEU: INTERRUPT VECTOR TABLE BASE ADDRESS UPPER REGISTER

|       |     |     |             |         |         |         |         |
|-------|-----|-----|-------------|---------|---------|---------|---------|
| U-0   | U-0 | U-0 | R/W-0/0     | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| —     | —   | —   | BASE<20:16> |         |         |         |         |
| bit 7 |     |     |             |         |         |         |         |
|       |     |     |             |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-5 **Unimplemented:** Read as '0'

bit 4-0 **BASE<20:16>:** Interrupt Vector Table Base Address bits

## REGISTER 9-34: IVTBASEH: INTERRUPT VECTOR TABLE BASE ADDRESS HIGH REGISTER

|            |         |         |         |         |         |         |         |
|------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0    | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| BASE<15:8> |         |         |         |         |         |         |         |
| bit 7      |         |         |         |         |         |         |         |
|            |         |         |         |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-0 **BASE<15:8>:** Interrupt Vector Table Base Address bits

## REGISTER 9-35: IVTBASEL: INTERRUPT VECTOR TABLE BASE ADDRESS LOW REGISTER

|           |         |         |         |         |         |         |         |
|-----------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0   | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-1/1 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| BASE<7:0> |         |         |         |         |         |         |         |
| bit 7     |         |         |         |         |         |         |         |
|           |         |         |         |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-0 **BASE<7:0>:** Interrupt Vector Table Base Address bits

## REGISTER 9-36: IVTADU: INTERRUPT VECTOR TABLE ADDRESS UPPER REGISTER

|       |     |     |           |       |       |       |       |
|-------|-----|-----|-----------|-------|-------|-------|-------|
| U-0   | U-0 | U-0 | R-0/0     | R-0/0 | R-0/0 | R-0/0 | R-0/0 |
| —     | —   | —   | AD<20:16> |       |       |       |       |
| bit 7 |     |     |           |       |       |       |       |
|       |     |     |           |       |       |       | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-5 **Unimplemented:** Read as '0'

bit 4-0 **AD<20:16>:** Interrupt Vector Table Address bits

## REGISTER 9-37: IVTADH: INTERRUPT VECTOR TABLE ADDRESS HIGH REGISTER

|          |       |       |       |       |       |       |       |
|----------|-------|-------|-------|-------|-------|-------|-------|
| R-0/0    | R-0/0 | R-0/0 | R-0/0 | R-0/0 | R-0/0 | R-0/0 | R-0/0 |
| AD<15:8> |       |       |       |       |       |       |       |
| bit 7    |       |       |       |       |       |       |       |
|          |       |       |       |       |       |       | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-0 **AD<15:8>:** Interrupt Vector Table Address bits

## REGISTER 9-38: IVTADL: INTERRUPT VECTOR TABLE ADDRESS LOW REGISTER

|         |       |       |       |       |       |       |       |
|---------|-------|-------|-------|-------|-------|-------|-------|
| R-0/0   | R-0/0 | R-0/0 | R-0/0 | R-1/1 | R-0/0 | R-0/0 | R-0/0 |
| AD<7:0> |       |       |       |       |       |       |       |
| bit 7   |       |       |       |       |       |       |       |
|         |       |       |       |       |       |       | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-0 **AD<7:0>:** Interrupt Vector Table Address bits

## REGISTER 9-39: IVTLOCK: INTERRUPT VECTOR TABLE LOCK REGISTER

|       |     |     |     |     |     |     |                            |
|-------|-----|-----|-----|-----|-----|-----|----------------------------|
| U-0   | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0/0                    |
| —     | —   | —   | —   | —   | —   | —   | IVTLOCKED <sup>(1,2)</sup> |
| bit 7 |     |     |     |     |     |     | bit 0                      |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-1 **Unimplemented:** Read as '0'

bit 0 **IVTLOCKED:** IVT Registers Lock bits<sup>(1,2)</sup>

1 = IVTBASE Registers are locked and cannot be written

0 = IVTBASE Registers can be modified by write operations

**Note 1:** The IVTLOCK bit can only be set or cleared after the unlock sequence in [Example 9-1](#).

**Note 2:** If IVT1WAY = 1, the IVTLOCK bit cannot be cleared after it has been set. See [Register 5-3](#).

## REGISTER 9-40: SHADCON: SHADOW CONTROL REGISTER

|       |     |     |     |     |     |     |         |
|-------|-----|-----|-----|-----|-----|-----|---------|
| U-0   | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0/0 |
| —     | —   | —   | —   | —   | —   | —   | SHADLO  |
| bit 7 |     |     |     |     |     |     | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-1 **Unimplemented:** Read as '0'

bit 0 **SHADLO:** Interrupt Shadow Register Access Switch bit

0 = Access Main Context for Interrupt Shadow Registers

1 = Access Low-Priority Interrupt Context for Interrupt Shadow Registers

# PIC18(L)F25/26K83

**TABLE 9-3: SUMMARY OF REGISTERS ASSOCIATED WITH INTERRUPTS**

| Name     | Bit 7      | Bit 6     | Bit 5      | Bit 4         | Bit 3     | Bit 2     | Bit 1         | Bit 0         | Register on Page    |
|----------|------------|-----------|------------|---------------|-----------|-----------|---------------|---------------|---------------------|
| INTCON0  | GIE/GIEH   | GIEL      | IPEN       | —             | —         | INT2EDG   | INT1EDG       | INT0EDG       | <a href="#">125</a> |
| INTCON1  | STAT<1:0>  |           | —          | —             | —         | —         | —             | —             | <a href="#">126</a> |
| PIE0     | IOCFIE     | CRCIE     | SCANIE     | NVMIE         | CSWIE     | OSFIE     | HLVDIE        | SWIE          | <a href="#">137</a> |
| PIE1     | SMT1PWAIE  | SMT1PRAIE | SMT1IE     | C1IE          | ADTIE     | ADIE      | ZCDIE         | INT0IE        | <a href="#">138</a> |
| PIE2     | I2C1RXIE   | SPI1IE    | SPI1TXIE   | SPI1RXIE      | DMA1AIE   | DMA1ORIE  | DMA1DCNTIE    | DMA1SCNTIE    | <a href="#">139</a> |
| PIE3     | TMR0IE     | U1IE      | U1EIE      | U1TXIE        | U1RXIE    | I2C1EIE   | I2C1IE        | I2C1TXIE      | <a href="#">140</a> |
| PIE4     | INT1IE     | CLC1IE    | CWG1IE     | NCO1IE        | CCP1IE    | TMR2IE    | TMR1GIE       | TMR1IFE       | <a href="#">141</a> |
| PIE5     | IRXIE      | WAKIE     | ERRIE      | TXB2IE/TXBnIE | TXB1IE    | TXB0IE    | RXB1IE/RXBnIE | RXB0IE/FIFOIE | <a href="#">142</a> |
| PIE6     | DMA2AIE    | DMA2ORIE  | DMA2DCNTIE | DMA2SCNTIE    | SMT2PWAIE | SMT2PRAIE | SMT2IE        | C2IE          | <a href="#">143</a> |
| PIE7     | U2IE       | U2EIE     | U2TXIE     | U2RXIE        | I2C2EIE   | I2C2IE    | I2C2TXIE      | I2C2RXIE      | <a href="#">144</a> |
| PIE8     | TMR5IE     | INT2IE    | CLC2IE     | CWG2IE        | CCP2IE    | TMR4IE    | TMR3GIE       | TMR3IE        | <a href="#">145</a> |
| PIE9     | —          | CLC4IE    | CCP4IE     | CLC3IE        | CWG3IE    | CCP3IE    | TMR6IE        | TMR5IE        | <a href="#">146</a> |
| PIR0     | IOCFIF     | CRCIF     | SCANIF     | NVMIF         | CSWIF     | OSFIF     | HLVDIF        | SWIF          | <a href="#">127</a> |
| PIR1     | SMT1PWAIF  | SMT1PRAIF | SMT1IF     | C1IF          | ADTIF     | ADIF      | ZCDIF         | INT0IF        | <a href="#">128</a> |
| PIR2     | I2C1RXIF   | SPI1IF    | SPI1TXIF   | SPI1RXIF      | DMA1AIF   | DMA1ORIF  | DMA1DCNTIF    | DMA1SCNTIF    | <a href="#">129</a> |
| PIR3     | TMR0IF     | U1IF      | U1EIF      | U1TXIF        | U1RXIF    | I2C1EIF   | I2C1IF        | I2C1TXIF      | <a href="#">130</a> |
| PIR4     | INT1IF     | CLC1IF    | CWG1IF     | NCO1IF        | CCP1IF    | TMR2IF    | TMR1GIF       | TMR1IF        | <a href="#">131</a> |
| PIR5     | IRXIF      | WAKIF     | ERRIF      | TXB2IF/TXBnIF | TXB1IF    | TXB0IF    | RXB1IF/RXBnIF | RXB0IF/FIFOIF | <a href="#">132</a> |
| PIR6     | DMA2AIF    | DMA2ORIF  | DMA2DCNTIF | DMA2SCNTIF    | SMT2PWAIF | SMT2PRAIF | SMT2IF        | C2IF          | <a href="#">133</a> |
| PIR7     | U2IF       | U2EIF     | U2TXIF     | U2RXIF        | I2C2EIF   | I2C2IF    | I2C2TXIF      | I2C2RXIF      | <a href="#">134</a> |
| PIR8     | TMR5IF     | INT2IF    | CLC2IF     | CWG2IF        | CCP2IF    | TMR4IF    | TMR3GIF       | TMR3IF        | <a href="#">135</a> |
| PIR9     | —          | CLC4IF    | CCP4IF     | CLC3IF        | CWG3IF    | CCP3IF    | TMR6IF        | TMR5IF        | <a href="#">136</a> |
| IPR0     | IOCIP      | CRCIP     | SCANIP     | NVMIP         | CSWIP     | OSFIP     | HLVDIP        | SWIP          | <a href="#">147</a> |
| IPR1     | SMT1PWAIP  | SMT1PRAIP | SMT1IP     | C1IP          | ADTIP     | ADIP      | ZCDIP         | INT0IP        | <a href="#">148</a> |
| IPR2     | I2C1RIP    | SPI1IP    | SPI1TIP    | SPI1RIP       | DMA1AIP   | DMA1ORIP  | DMA1DCNTIP    | DMA1SCNTIP    | <a href="#">149</a> |
| IPR3     | TMR0IP     | U1IP      | U1EIP      | U1TXIP        | U1RXIP    | I2C1EIP   | I2C1IP        | I2C1TXIP      | <a href="#">150</a> |
| IPR4     | INT1IP     | CLC1IP    | CWG1IP     | NCO1IP        | CCP1IP    | TMR2IP    | TMR1GIP       | TMR1IP        | <a href="#">151</a> |
| IPR5     | IRXIP      | WAKIP     | ERRIP      | TXB2IP/TXBnIP | TXB1IP    | TXB0IP    | RXB1IP/RXBnIP | RXB0IP/FIFOIP | <a href="#">152</a> |
| IPR6     | DMA2AIP    | DMA2ORIP  | DMA2DCNTIP | DMA2SCNTIP    | SMT2PWAIP | SMT2PRAIP | SMT2IP        | C2IP          | <a href="#">153</a> |
| IPR7     | U2IP       | U2EIP     | U2TXIP     | U2RXIP        | I2C2EIP   | I2C2IP    | I2C2TXIP      | I2C2RXIP      | <a href="#">154</a> |
| IPR8     | TMR5IP     | INT2IP    | CLC2IP     | CWG2IP        | CCP2IP    | TMR4IP    | TMR3GIP       | TMR3IP        | <a href="#">155</a> |
| IPR9     | —          | CLC4IP    | CCP4IP     | CLC3IP        | CWG3IP    | CCP3IP    | TMR6IP        | TMR5IP        | <a href="#">156</a> |
| IVTBASEU | —          | —         | —          | BASE<20:16>   |           |           |               |               | <a href="#">157</a> |
| IVTBASEH | BASE<15:8> |           |            |               |           |           |               |               | <a href="#">157</a> |
| IVTBASEL | BASE<7:0>  |           |            |               |           |           |               |               | <a href="#">157</a> |
| IVTADU   | —          | —         | —          | AD<20:16>     |           |           |               |               | <a href="#">158</a> |
| IVTADH   | AD<15:8>   |           |            |               |           |           |               |               | <a href="#">158</a> |
| IVTADL   | AD<7:0>    |           |            |               |           |           |               |               | <a href="#">158</a> |
| IVTLOCK  | —          | —         | —          | —             | —         | —         | —             | IVTLOCKED     | <a href="#">159</a> |

**Legend:** — = unimplemented locations, read as '0'. Shaded bits are not used for interrupts.



## 10.0 POWER-SAVING OPERATION MODES

The purpose of the Power-Down modes is to reduce power consumption. There are three Power-Down modes:

- Doze mode
- Sleep mode
- Idle mode

### 10.1 Doze Mode

Doze mode allows for power saving by reducing CPU operation and program memory (PFM) access, without affecting peripheral operation. Doze mode differs from Sleep mode because the bandgap and system oscillators continue to operate, while only the CPU and PFM are affected. The reduced execution saves power by eliminating unnecessary operations within the CPU and memory.

When the Doze Enable (DOZEN) bit is set (DOZEN = 1), the CPU executes only one instruction cycle out of every N cycles as defined by the DOZE<2:0> bits of the CPUDOZE register. For example, if DOZE<2:0> = 001, the instruction cycle

ratio is 1:4. The CPU and memory execute for one instruction cycle and then lay idle for three instruction cycles. During the unused cycles, the peripherals continue to operate at the system clock speed.

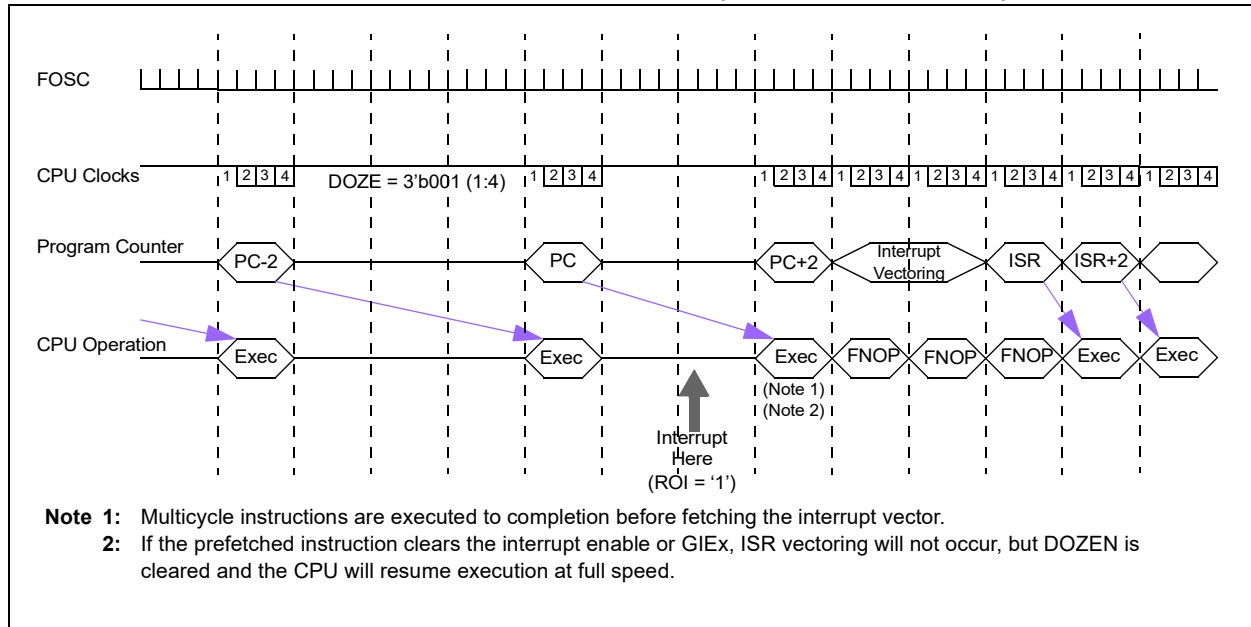
#### 10.1.1 DOZE OPERATION

The Doze operation is illustrated in Figure 10-1. For this example:

- Doze enable (DOZEN) bit set (DOZEN = 1)
- DOZE<2:0> = 001 (1:4) ratio
- Recover-on-Interrupt (ROI) bit set (ROI = 1)

As with normal operation, the PFM fetches for the next instruction cycle. The Q-clocks to the peripherals continue throughout.

**FIGURE 10-1: DOZE MODE OPERATION EXAMPLE (DOZE<2:0> = 001, 1:4)**



## 10.1.2 INTERRUPTS DURING DOZE

If an interrupt occurs and the Recover-On-Interrupt bit is clear (ROI = 0) at the time of the interrupt, the Interrupt Service Routine (ISR) continues to execute at the rate selected by DOZE<2:0>. Interrupt latency is extended by the DOZE<2:0> ratio.

If an interrupt occurs and the ROI bit is set (ROI = 1) at the time of the interrupt, the DOZEN bit is cleared and the CPU executes at full speed. The prefetched instruction is executed and then the interrupt vector sequence is executed. In [Figure 10-1](#), the interrupt occurs during the 2<sup>nd</sup> instruction cycle of the Doze period, and immediately brings the CPU out of Doze. If the Doze-On-Exit (DOE) bit is set (DOE = 1) when the RETFIE operation is executed, DOZEN is set, and the CPU executes at the reduced rate based on the DOZE<2:0> ratio.

### EXAMPLE 10-1: DOZE SOFTWARE EXAMPLE

```
//Mainline operation
bool somethingToDo = FALSE;
void main()
{
    initializeSystem();
        // DOZE = 64:1 (for example)
        // ROI = 1;
    GIE = 1; // enable interrupts
    while (1)
    {
        // If ADC completed, process data
        if (somethingToDo)
        {
            doSomething();
            DOZEN = 1; // resume low-power
        }
    }
}

// Data interrupt handler
void interrupt()
{
    // DOZEN = 0 because ROI = 1
    if (ADIF)
    {
        somethingToDo = TRUE;
        DOE = 0; // make main() go fast
        ADIF = 0;
    }
    // else check other interrupts...
    if (TMR0IF)
    {
        timerTick++;
        DOE = 1; // make main() go slow
        TMR0IF = 0;
    }
}
```

## 10.2 Sleep Mode

Sleep mode is entered by executing the SLEEP instruction, while the Idle Enable (IDLEN) bit of the CPUDOZE register is clear (IDLEN = 0).

Upon entering Sleep mode, the following conditions exist:

1. WDT will be cleared but keeps running if enabled for operation during Sleep
2. The  $\overline{PD}$  bit of the STATUS register is cleared ([Register 4-2](#))
3. The  $\overline{TO}$  bit of the STATUS register is set ([Register 4-2](#))
4. The CPU clock is disabled
5. LFINTOSC, SOSC, HFINTOSC and ADCRC are unaffected and peripherals using them may continue operation in Sleep.
6. I/O ports maintain the status they had before Sleep was executed (driving high, low, or high-impedance)
7. Resets other than WDT are not affected by Sleep mode

Refer to individual chapters for more details on peripheral operation during Sleep.

To minimize current consumption, the following conditions should be considered:

- I/O pins should not be floating
- External circuitry sinking current from I/O pins
- Internal circuitry sourcing current from I/O pins
- Current draw from pins with internal weak pull-ups
- Modules using any oscillator

I/O pins that are high-impedance inputs should be pulled to VDD or VSS externally to avoid switching currents caused by floating inputs.

Examples of internal circuitry that might be sourcing current include modules such as the DAC and FVR modules. See [Section 38.0 “5-Bit Digital-to-Analog Converter \(DAC\) Module”](#) and [Section 35.0 “Fixed Voltage Reference \(FVR\)”](#) for more information on these modules.

## 10.2.1 WAKE-UP FROM SLEEP

The device can wake up from Sleep through one of the following events:

1. External Reset input on  $\overline{\text{MCLR}}$  pin, if enabled
2. BOR Reset, if enabled
3. Low-Power Brown-Out Reset (LPBOR), if enabled
4. POR Reset
5. Windowed Watchdog Timer, if enabled
6. All interrupt sources except clock switch interrupt can wake up the part.

The first five events will cause a device Reset. The last one event is considered a continuation of program execution. To determine whether a device Reset or wake-up event occurred, refer to [Section 6.13 “Power Control \(PCON0/PCON1\) Register”](#).

When the `SLEEP` instruction is being executed, the next instruction (`PC + 2`) is prefetched. For the device to wake-up through an interrupt event, the corresponding Interrupt Enable bit must be enabled. Wake-up will occur regardless of the state of the GIE bit. If the GIE bit is disabled, the device continues execution at the instruction after the `SLEEP` instruction. If the GIE bit is enabled, the device executes the instruction after the `SLEEP` instruction, the device will then call the Interrupt Service Routine. In cases where the execution of the instruction following `SLEEP` is not desirable, the user should have a `NOP` after the `SLEEP` instruction.

The WDT is cleared when the device wakes-up from Sleep, regardless of the source of wake-up.

Upon a wake from a Sleep event, the core will wait for a combination of three conditions before beginning execution. The conditions are:

- PFM Ready
- COSC-Selected Oscillator Ready
- BOR Ready (unless BOR is disabled)

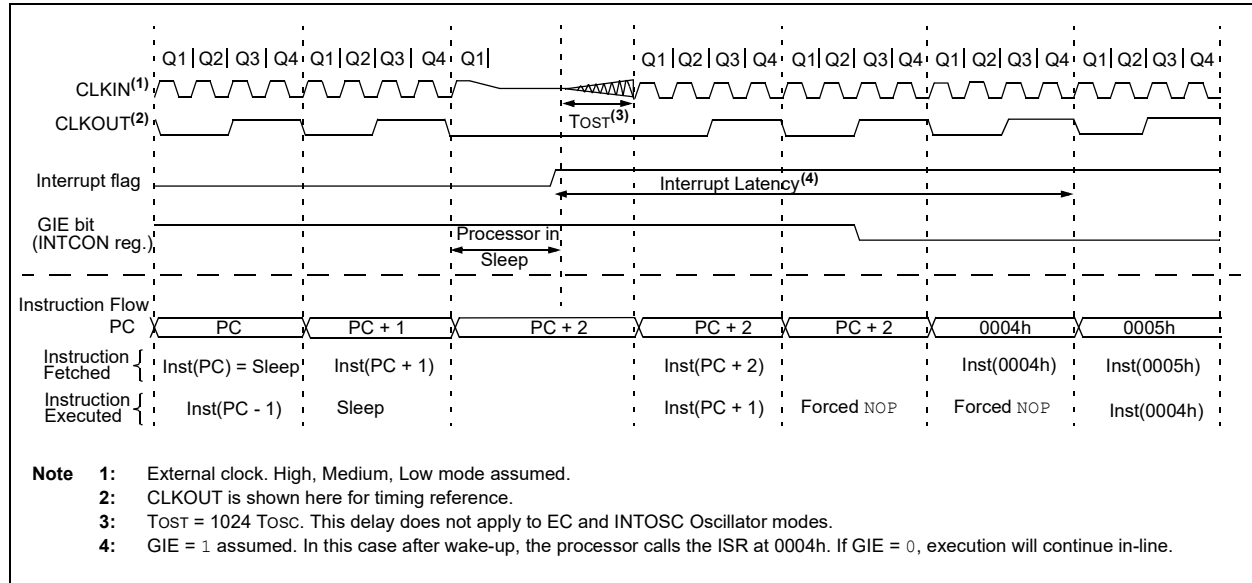
## 10.2.2 WAKE-UP USING INTERRUPTS

When any interrupt source, with the exception of the clock switch interrupt, has both its interrupt enable bit and interrupt flag bit set, one of the following will occur:

- If the interrupt occurs **before** the execution of a `SLEEP` instruction
  - `SLEEP` instruction will execute as a `NOP`
  - WDT and WDT prescaler will not be cleared
  - $\overline{\text{TO}}$  bit of the STATUS register will not be set
  - $\overline{\text{PD}}$  bit of the STATUS register will not be cleared
- If the interrupt occurs **during or after** the execution of a `SLEEP` instruction
  - `SLEEP` instruction will be completely executed
  - Device will immediately wake-up from Sleep
  - WDT and WDT prescaler will be cleared
  - $\overline{\text{TO}}$  bit of the STATUS register will be set
  - $\overline{\text{PD}}$  bit of the STATUS register will be cleared

Even if the flag bits were checked before executing a `SLEEP` instruction, it may be possible for flag bits to become set before the `SLEEP` instruction completes. To determine whether a `SLEEP` instruction executed, test the  $\overline{\text{PD}}$  bit. If the  $\overline{\text{PD}}$  bit is set, the `SLEEP` instruction was executed as a `NOP`.

**FIGURE 10-2: WAKE-UP FROM SLEEP THROUGH INTERRUPT**



## 10.2.3 LOW-POWER SLEEP MODE

The PIC18F25/26K83 device family contains an internal Low Dropout (LDO) voltage regulator, which allows the device I/O pins to operate at voltages up to 5.5V while the internal device logic operates at a lower voltage. The LDO and its associated reference circuitry must remain active when the device is in Sleep mode.

The PIC18F25/26K83 devices allows the user to optimize the operating current in Sleep, depending on the application requirements.

Low-Power Sleep mode can be selected by setting the VREGPM bit of the VREGCON register.

### 10.2.3.1 Sleep Current vs. Wake-up Time

In the default operating mode, the LDO and reference circuitry remain in the normal configuration while in Sleep. The device is able to exit Sleep mode quickly since all circuits remain active. In Low-Power Sleep mode, when waking-up from Sleep, an extra delay time is required for these circuits to return to the normal configuration and stabilize.

The Low-Power Sleep mode is beneficial for applications that stay in Sleep mode for long periods of time. The Normal mode is beneficial for applications that need to wake from Sleep quickly and frequently.

## 10.2.3.2 Peripheral Usage in Sleep

Some peripherals that can operate in Sleep mode will not operate properly with the Low-Power Sleep mode selected. The Low-Power Sleep mode is intended for use with these peripherals:

- Brown-out Reset (BOR)
- Windowed Watchdog Timer (WWDT)
- External interrupt pin/Interrupt-On-Change pins
- Peripherals that run off external secondary clock source

It is the responsibility of the end user to determine what is acceptable for their application when setting the VREGPM settings in order to ensure operation in Sleep.

**Note:** The PIC18LF25/26K83 devices do not have a configurable Low-Power Sleep mode. PIC18LF25/26K83 devices are unregulated and are always in the lowest power state when in Sleep, with no wake-up time penalty. These devices have a lower maximum VDD and I/O voltage than the PIC18F25/26K83 devices. See [Section 45.0 “Electrical Specifications”](#) for more information.

## 10.2.4 IDLE MODE

When IDLEN is set (IDLEN = 1), the SLEEP instruction will put the device into Idle mode. In Idle mode, the CPU and memory operations are halted, but the peripheral clocks continue to run. This mode is similar to Doze mode, except that in IDLE both the CPU and PFM are shut off.

**Note:** If CLKOUTEN is enabled (CLKOUTEN = 0, Configuration Word 1H), the output will continue operating while in Idle.

## 10.2.4.1 Idle and Interrupts

IDLE mode ends when an interrupt occurs (even if GIE = 0), but IDLEN is not changed. The device can re-enter IDLE by executing the SLEEP instruction.

If Recover-On-Interrupt is enabled (ROI = 1), the interrupt that brings the device out of Idle also restores full-speed CPU execution when doze is also enabled.

## 10.2.4.2 Idle and WWDT

When in Idle, the WWDT Reset is blocked and will instead wake the device. The WWDT wake-up is not an interrupt, therefore ROI does not apply.

**Note:** The WDT can bring the device out of Idle, in the same way it brings the device out of Sleep. The DOZEN bit is not affected.

## 10.3 Peripheral Operation in Power Saving Modes

All selected clock sources and the peripherals running off them are active in both IDLE and DOZE mode. Only in Sleep mode, both the FOSC and FOSC/4 clocks are unavailable. All the other clock sources are active, if enabled manually or through peripheral clock selection before the part enters Sleep.

## 10.4 Register Definitions: Voltage Regulator Control

### REGISTER 10-1: VREGCON: VOLTAGE REGULATOR CONTROL REGISTER<sup>(1)</sup>

| U-0   | U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-1/1  |
|-------|-----|-----|-----|-----|-----|---------|----------|
| —     | —   | —   | —   | —   | —   | VREGPM  | Reserved |
| bit 7 |     |     |     |     |     |         | bit 0    |

#### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-2 **Unimplemented:** Read as '0'

bit 1 **VREGPM:** Voltage Regulator Power Mode Selection bit

1 = Low-Power Sleep mode enabled in Sleep<sup>(2)</sup>

Draws lowest current in Sleep, slower wake-up

0 = Normal Power mode enabled in Sleep<sup>(2)</sup>

Draws higher current in Sleep, faster wake-up

bit 0 **Reserved:** Read as '1'. Maintain this bit set.

**Note 1:** Not present in LF parts.

**2:** See [Section 45.0 "Electrical Specifications"](#).

# PIC18(L)F25/26K83

**REGISTER 10-2: CPUDOZE: DOZE AND IDLE REGISTER**

| R/W-0/u | R/W/HC/HS-0/0 | R/W-0/0 | R/W-0/0 | U-0 | R/W-0/0   | R/W-0/0 | R/W-0/0 |
|---------|---------------|---------|---------|-----|-----------|---------|---------|
| IDLEN   | DOZEN         | ROI     | DOE     | —   | DOZE<2:0> |         |         |
| bit 7   |               |         |         |     |           |         | bit 0   |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

HC = Bit is cleared by hardware

bit 7

**IDLEN:** Idle Enable bit

1 = A *SLEEP* instruction inhibits the CPU clock, but not the peripheral clock(s)

0 = A *SLEEP* instruction places the device into full Sleep mode

bit 6

**DOZEN:** Doze Enable bit<sup>(1,2)</sup>

1 = The CPU executes instruction cycles according to DOZE setting

0 = The CPU executes all instruction cycles (fastest, highest power operation)

bit 5

**ROI:** Recover-On-Interrupt bit

1 = Entering the Interrupt Service Routine (ISR) makes DOZEN = 0 bit, bringing the CPU to full-speed operation

0 = Interrupt entry does not change DOZEN

bit 4

**DOE:** Doze-On-Exit bit

1 = Executing RETFIE makes DOZEN = 1, bringing the CPU to reduced speed operation

0 = RETFIE does not change DOZEN

bit 3

**Unimplemented:** Read as '0'

bit 2-0

**DOZE<2:0>:** Ratio of CPU Instruction Cycles to Peripheral Instruction Cycles

111 =1:256

110 =1:128

101 =1:64

100 =1:32

011 =1:16

010 =1:8

001 =1:4

000 =1:2

**Note 1:** When ROI = 1 or DOE = 1, DOZEN is changed by hardware interrupt entry and/or exit.

**Note 2:** Entering ICD overrides DOZEN, returning the CPU to full execution speed; this bit is not affected.

**TABLE 10-1: SUMMARY OF REGISTERS ASSOCIATED WITH POWER-DOWN MODE**

| Name                   | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2     | Bit 1  | Bit 0    | Register on Page |
|------------------------|-------|-------|-------|-------|-------|-----------|--------|----------|------------------|
| VREGCON <sup>(†)</sup> | —     | —     | —     | —     | —     | —         | VREGPM | Reserved | 166              |
| CPUDOZE                | IDLEN | DOZEN | ROI   | DOE   | —     | DOZE<2:0> |        |          | 167              |

**Legend:** — = unimplemented location, read as '0'. Shaded cells are not used in Power-Down mode.

**Note 1:** Not present in LF parts.

## 11.0 WINDOWED WATCHDOG TIMER (WWDT)

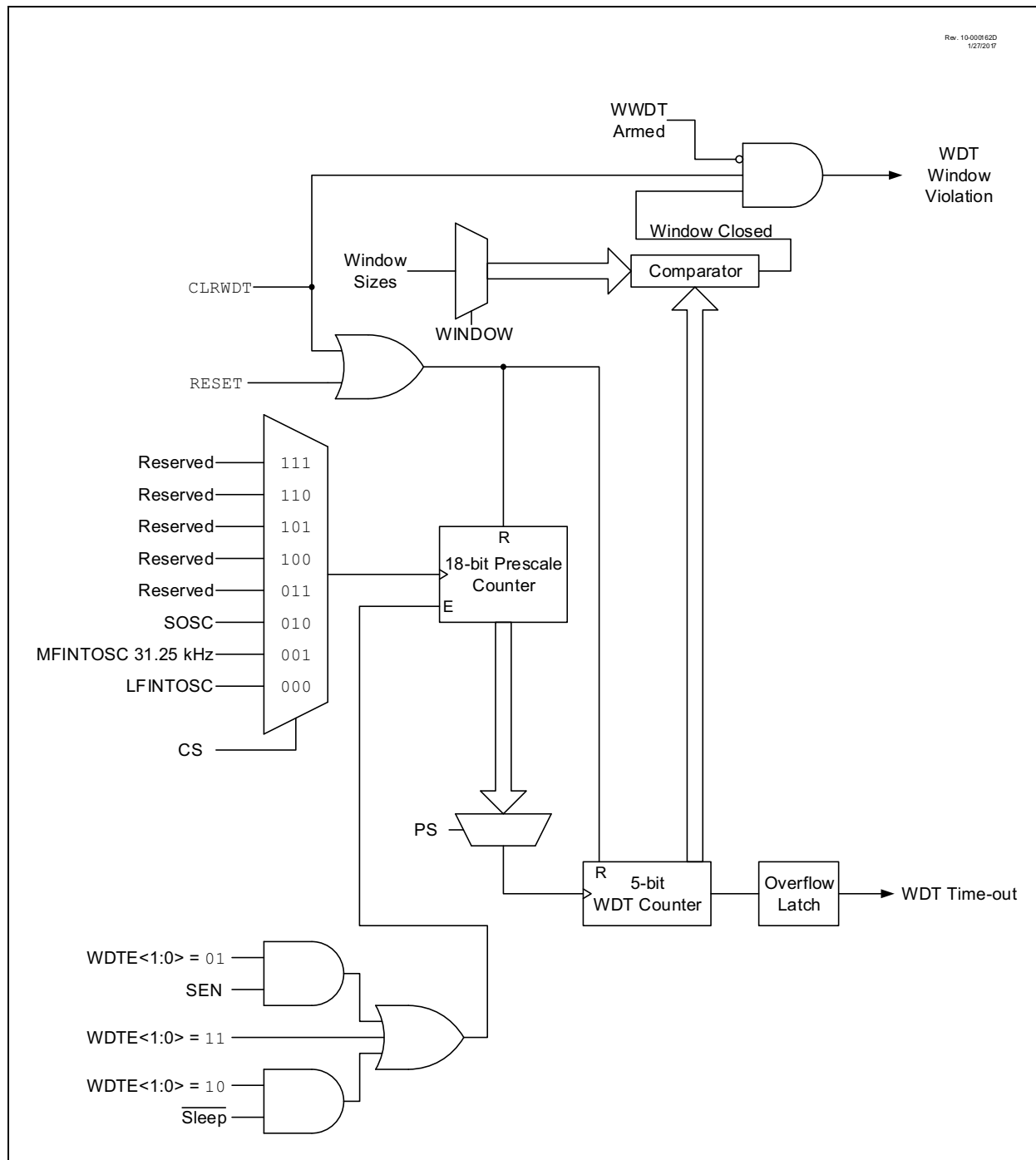
The Watchdog Timer (WDT) is a system timer that generates a Reset if the firmware does not issue a `CLRWDT` instruction within the time-out period. The Watchdog Timer is typically used to recover the system from unexpected events. The Windowed Watchdog Timer (WWDT) differs in that `CLRWDT` instructions are only accepted when they are performed within a specific window during the time-out period.

The WWDT has the following features:

- Selectable clock source
- Multiple operating modes
  - WWDT is always On
  - WWDT is off when in Sleep
  - WWDT is controlled by software
  - WWDT is always Off
- Configurable time-out period is from 1 ms to 256s (nominal)
- Configurable window size from 12.5% to 100% of the time-out period
- Multiple Reset conditions



**FIGURE 11-1: WINDOWED WATCHDOG TIMER BLOCK DIAGRAM**



## 11.1 Independent Clock Source

The WWDT can derive its time base from either the 31 kHz LFINTOSC or 31.25 kHz MFINTOSC internal oscillators, depending on the value of WDTE<1:0> Configuration bits.

If WDTE = 0b1x, then the clock source will be enabled depending on the WDTCCS<2:0> Configuration bits.

If WDTE = 0b01, the SEN bit should be set by software to enable WWDT, and the clock source is enabled by the CS bits in the WDTCON1 register.

Time intervals in this chapter are based on a minimum nominal interval of 1 ms. See [Section 45.0 “Electrical Specifications”](#) for LFINTOSC and MFINTOSC tolerances.

## 11.2 WWDT Operating Modes

The Windowed Watchdog Timer module has four operating modes controlled by the WDTE<1:0> bits in Configuration Words. See [Table 11-1](#).

### 11.2.1 WWDT IS ALWAYS ON

When the WDTE bits of Configuration Words are set to ‘11’, the WWDT is always on.

WWDT protection is active during Sleep.

### 11.2.2 WWDT IS OFF IN SLEEP

When the WDTE bits of Configuration Words are set to ‘10’, the WWDT is on, except in Sleep.

WWDT protection is not active during Sleep.

### 11.2.3 WWDT CONTROLLED BY SOFTWARE

When the WDTE bits of Configuration Words are set to ‘01’, the WWDT is controlled by the SEN bit of the WDTCON0 register.

WWDT protection is unchanged by Sleep. See [Table 11-1](#) for more details.

**TABLE 11-1: WWDT OPERATING MODES**

| WDTE<1:0> | SEN | Device Mode | WWDT Mode |
|-----------|-----|-------------|-----------|
| 11        | X   | X           | Active    |
| 10        | X   | Awake       | Active    |
|           |     | Sleep       | Disabled  |
| 01        | 1   | X           | Active    |
|           | 0   | X           | Disabled  |
| 00        | X   | X           | Disabled  |

## 11.3 Time-out Period

If the WDTCPSS<4:0> Configuration bits default to 0b11111, then the PS bits of the WDTCON0 register set the time-out period from 1 ms to 256 seconds (nominal). If any value other than the default value is assigned to WDTCPSS<4:0> Configuration bits, then the timer period will be based on the WDTCPSS<4:0> bits in the CONFIG3L register. After a Reset, the default time-out period is 2s.

## 11.4 Watchdog Window

The Windowed Watchdog Timer has an optional Windowed mode that is controlled by the WDTWWS<2:0> Configuration bits and WINDOW<2:0> bits of the WDTCON1 register. In the Windowed mode, the CLRWDT instruction must occur within the allowed window of the WDT period. Any CLRWDT instruction that occurs outside of this window will trigger a window violation and will cause a WWDT Reset, similar to a WWDT time out. See [Figure 11-2](#) for an example.

The window size is controlled by the WINDOW<2:0> Configuration bits, or the WINDOW<2:0> bits of WDTCON1, if WDTWWS<2:0> = 111.

The five Most Significant bits of the WDTTMR register are used to determine whether the window is open, as defined by the WINDOW<2:0> bits of the WDTCON1 register.

In the event of a window violation, a Reset will be generated and the WDTWV bit of the PCON0 register will be cleared. This bit is set by a POR or can be set in firmware.

## 11.5 Clearing the WWDT

The WWDT is cleared when any of the following conditions occur:

- Any Reset
- Valid CLRWDT instruction is executed
- Device enters Sleep
- Exit Sleep by Interrupt
- WWDT is disabled
- Oscillator Start-up Timer (OST) is running
- Any write to the WDTCON0 or WDTCON1 registers

### 11.5.1 CLRWDT CONSIDERATIONS (WINDOWED MODE)

When in Windowed mode, the WWDT must be armed before a CLRWDT instruction will clear the timer. This is performed by reading the WDTCON0 register. Executing a CLRWDT instruction without performing such an arming action will trigger a window violation regardless of whether the window is open or not.

See [Table 11-2](#) for more information.

## 11.6 Operation During Sleep

When the device enters Sleep, the WWDT is cleared. If the WWDT is enabled during Sleep, the WWDT resumes counting. When the device exits Sleep, the WWDT is cleared again.

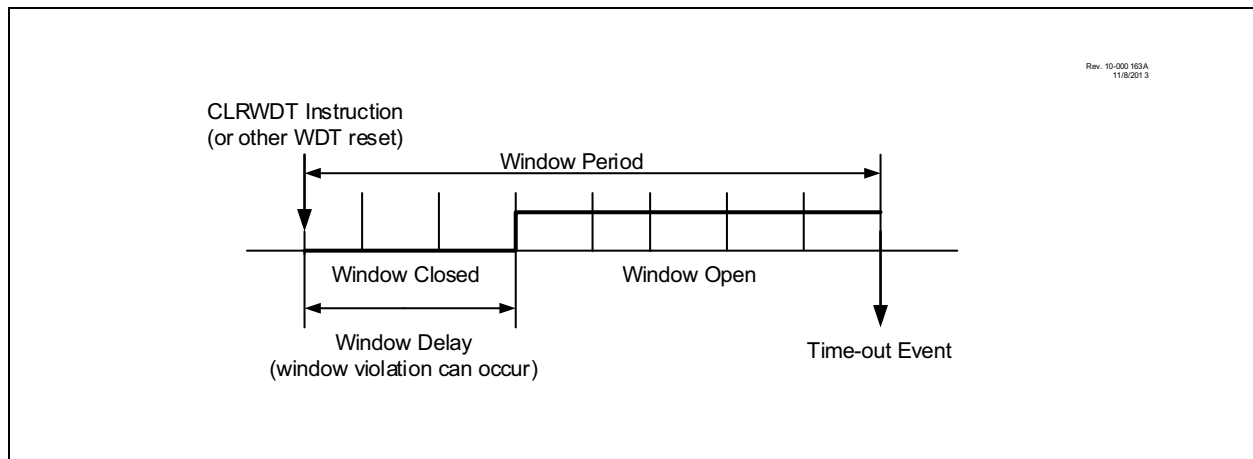
The WWDT remains clear until the Oscillator Start-up Timer (OST) completes, if enabled. See [Section 7.2.1.3 “Oscillator Start-up Timer \(OST\)”](#) for more information on the OST.

When a WWDT time-out occurs while the device is in Sleep, no Reset is generated. Instead, the device wakes up and resumes operation. The  $\overline{TO}$  and  $\overline{PD}$  bits in the STATUS register are changed to indicate the event. The RWDT bit in the PCON0 register can also be used. See [Section 4.0 “Memory Organization”](#) for more information.

**TABLE 11-2: WWDT CLEARING CONDITIONS**

| Conditions                                              | WWDT                         |
|---------------------------------------------------------|------------------------------|
| WDTE<1:0> = 00                                          | Cleared                      |
| WDTE<1:0> = 01 and SEN = 0                              |                              |
| WDTE<1:0> = 10 and enter Sleep                          |                              |
| CLRWDT Command                                          |                              |
| Oscillator Fail Detected                                |                              |
| Exit Sleep + System Clock = SOSC, EXTRC, INTOSC, EXTCLK |                              |
| Exit Sleep + System Clock = XT, HS, LP                  | Cleared until the end of OST |
| Change INTOSC divider (IRCF bits)                       | Unaffected                   |

**FIGURE 11-2: WINDOW PERIOD AND DELAY**



## 11.7 Register Definitions: Windowed Watchdog Timer Control

**REGISTER 11-1: WDTCON0: WATCHDOG TIMER CONTROL REGISTER 0**

| U-0   | U-0 | R/W <sup>(3)</sup> -q/q <sup>(2)</sup> | R/W <sup>(3)</sup> -q/q <sup>(2)</sup> | R/W <sup>(3)</sup> -q/q <sup>(2)</sup> | R/W <sup>(3)</sup> -q/q <sup>(2)</sup> | R/W <sup>(3)</sup> -q/q <sup>(2)</sup> | R/W-0/0 |
|-------|-----|----------------------------------------|----------------------------------------|----------------------------------------|----------------------------------------|----------------------------------------|---------|
| —     | —   | PS<4:0>                                |                                        |                                        |                                        |                                        | SEN     |
| bit 7 |     |                                        |                                        |                                        |                                        |                                        | bit 0   |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

q = Value depends on condition

bit 7-6 **Unimplemented:** Read as '0'

bit 5-1 **PS<4:0>:** Watchdog Timer Prescale Select bits<sup>(1)</sup>

Bit Value = Prescale Rate

11111 = Reserved. Results in minimum interval (1:32)

•

•

•

10011 = Reserved. Results in minimum interval (1:32)

10010 = 1:8388608 (2<sup>23</sup>) (Interval 256s nominal)

10001 = 1:4194304 (2<sup>22</sup>) (Interval 128s nominal)

10000 = 1:2097152 (2<sup>21</sup>) (Interval 64s nominal)

01111 = 1:1048576 (2<sup>20</sup>) (Interval 32s nominal)

01110 = 1:524288 (2<sup>19</sup>) (Interval 16s nominal)

01101 = 1:262144 (2<sup>18</sup>) (Interval 8s nominal)

01100 = 1:131072 (2<sup>17</sup>) (Interval 4s nominal)

01011 = 1:65536 (Interval 2s nominal) (Reset value)

01010 = 1:32768 (Interval 1s nominal)

01001 = 1:16384 (Interval 512 ms nominal)

01000 = 1:8192 (Interval 256 ms nominal)

00111 = 1:4096 (Interval 128 ms nominal)

00110 = 1:2048 (Interval 64 ms nominal)

00101 = 1:1024 (Interval 32 ms nominal)

00100 = 1:512 (Interval 16 ms nominal)

00011 = 1:256 (Interval 8 ms nominal)

00010 = 1:128 (Interval 4 ms nominal)

00001 = 1:64 (Interval 2 ms nominal)

00000 = 1:32 (Interval 1 ms nominal)

bit 0 **SEN:** Software Enable/Disable for Watchdog Timer bit

If WDTE<1:0> = 1x:

This bit is ignored.

If WDTE<1:0> = 01:

1 = WDT is turned on

0 = WDT is turned off

If WDTE<1:0> = 00:

This bit is ignored.

**Note 1:** Times are approximate. WDT time is based on 31 kHz LFINTOSC.

**2:** When WDTCP5 <4:0> in CONFIG3L = 11111, the Reset value of PS<4:0> is 01011. Otherwise, the Reset value of PS<4:0> is equal to WDTCP5<4:0> in CONFIG3L.

**3:** When WDTCP5 <4:0> in CONFIG3L ≠ 11111, these bits are read-only.

**4:** When the WWDT is configured to run using the SOSC as a clock source and the device is allowed to undergo a Reset, as triggered by a WDT time-out, the SOSC would also undergo a Reset. That means the SOSC will execute its start-up sequence which requires 1024 SOSC clock counts before it is made available for peripherals to use. So for example, if the WDT is set for a 1 ms time-out and the device is allowed to undergo a WDT Reset, then the actual WDT Reset period will be: WDT\_PERIOD = (1/(SOSC\_FREQUENCY) \* 1024) + 1 ms.

## REGISTER 11-2: WDTCON1: WATCHDOG TIMER CONTROL REGISTER 1

| U-0   | R/W <sup>(3)</sup> -q/q <sup>(1)</sup> | R/W <sup>(3)</sup> -q/q <sup>(1)</sup> | R/W <sup>(3)</sup> -q/q <sup>(1)</sup> | U-0 | R/W <sup>(4)</sup> -q/q <sup>(2)</sup> | R/W <sup>(4)</sup> -q/q <sup>(2)</sup> | R/W <sup>(4)</sup> -q/q <sup>(2)</sup> |
|-------|----------------------------------------|----------------------------------------|----------------------------------------|-----|----------------------------------------|----------------------------------------|----------------------------------------|
| —     | CS<2:0>                                |                                        |                                        | —   | WINDOW<2:0>                            |                                        |                                        |
| bit 7 |                                        |                                        |                                        |     |                                        |                                        | bit 0                                  |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

q = Value depends on condition

bit 7 **Unimplemented:** Read as '0'

bit 6-4 **CS<2:0>:** Watchdog Timer Clock Select bits

111 = Reserved

•

•

•

011 = Reserved

010 = SOSC

001 = MFINTOSC 31.25 kHz

000 = LFINTOSC 31 kHz

bit 3 **Unimplemented:** Read as '0'

bit 2-0 **WINDOW<2:0>:** Watchdog Timer Window Select bits

| WINDOW<2:0> | Window delay<br>Percent of time | Window opening<br>Percent of time |
|-------------|---------------------------------|-----------------------------------|
| 111         | N/A                             | 100                               |
| 110         | 12.5                            | 87.5                              |
| 101         | 25                              | 75                                |
| 100         | 37.5                            | 62.5                              |
| 011         | 50                              | 50                                |
| 010         | 62.5                            | 37.5                              |
| 001         | 75                              | 25                                |
| 000         | 87.5                            | 12.5                              |

**Note 1:** If WDTCCS <2:0> in CONFIG3H = 111, the Reset value of CS<2:0> is 000.

**2:** The Reset value of WINDOW<2:0> is determined by the value of WDTCCS<2:0> in the CONFIG3H register.

**3:** If WDTCCS<2:0> in CONFIG3H ≠ 111, these bits are read-only.

**4:** If WDTCCS<2:0> in CONFIG3H ≠ 111, these bits are read-only.

## REGISTER 11-3: WDTPSL: WWDT PRESCALE SELECT LOW BYTE REGISTER (READ-ONLY)

|            |       |       |       |       |       |       |       |
|------------|-------|-------|-------|-------|-------|-------|-------|
| R-0/0      | R-0/0 | R-0/0 | R-0/0 | R-0/0 | R-0/0 | R-0/0 | R-0/0 |
| PSCNT<7:0> |       |       |       |       |       |       |       |
| bit 7      |       |       |       | bit 0 |       |       |       |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-0 **PSCNT<7:0>**: Prescale Select Low Byte bits<sup>(1)</sup>

**Note 1:** The 18-bit WDT prescale value, PSCNT<17:0> includes the WDTPSL, WDTPSH and the lower bits of the WDTTMR registers. PSCNT<17:0> is intended for debug operations and should not be read during normal operation.

## REGISTER 11-4: WDTPSH: WWDT PRESCALE SELECT HIGH BYTE REGISTER (READ-ONLY)

|             |       |       |       |       |       |       |       |
|-------------|-------|-------|-------|-------|-------|-------|-------|
| R-0/0       | R-0/0 | R-0/0 | R-0/0 | R-0/0 | R-0/0 | R-0/0 | R-0/0 |
| PSCNT<15:8> |       |       |       |       |       |       |       |
| bit 7       |       |       |       | bit 0 |       |       |       |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-0 **PSCNT<15:8>**: Prescale Select High Byte bits<sup>(1)</sup>

**Note 1:** The 18-bit WDT prescale value, PSCNT<17:0> includes the WDTPSL, WDTPSH and the lower bits of the WDTTMR registers. PSCNT<17:0> is intended for debug operations and should not be read during normal operation.

## REGISTER 11-5: WDTTMR: WDT TIMER REGISTER (READ-ONLY)

|             |       |       |       |       |       |              |       |
|-------------|-------|-------|-------|-------|-------|--------------|-------|
| R-0/0       | R-0/0 | R-0/0 | R-0/0 | R-0/0 | R-0/0 | R-0/0        | R-0/0 |
| WDTTMR<4:0> |       |       |       |       | STATE | PSCNT<17:16> |       |
| bit 7       |       |       |       |       |       |              | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-3 **WDTTMR<4:0>**: Watchdog Window Value bits

| WINDOW | WDT Window State |             | Open Percent |
|--------|------------------|-------------|--------------|
|        | Closed           | Open        |              |
| 111    | N/A              | 00000-11111 | 100          |
| 110    | 00000-00011      | 00100-11111 | 87.5         |
| 101    | 00000-00111      | 01000-11111 | 75           |
| 100    | 00000-01011      | 01100-11111 | 62.5         |
| 011    | 00000-01111      | 10000-11111 | 50           |
| 010    | 00000-10011      | 10100-11111 | 37.5         |
| 001    | 00000-10111      | 11000-11111 | 25           |
| 000    | 00000-11011      | 11100-11111 | 12.5         |

bit 2 **STATE**: WDT Armed Status bit

1 = WDT is armed

0 = WDT is not armed

bit 1-0 **PSCNT<17:16>**: Prescale Select Upper Byte bits<sup>(1)</sup>

**Note 1:** The 18-bit WDT prescale value, PSCNT<17:0> includes the WDTPSL, WDTPSH and the lower bits of the WDTTMR registers. PSCNT<17:0> is intended for debug operations and should not be read during normal operation.

**TABLE 11-3: SUMMARY OF REGISTERS ASSOCIATED WITH WINDOWED WATCHDOG TIMER**

| Name    | Bit 7       | Bit 6   | Bit 5   | Bit 4 | Bit 3 | Bit 2       | Bit 1        | Bit 0 | Register on Page    |                     |
|---------|-------------|---------|---------|-------|-------|-------------|--------------|-------|---------------------|---------------------|
| WDTCON0 | —           | —       | PS<4:0> |       |       |             |              |       | SEN                 | <a href="#">172</a> |
| WDTCON1 | —           | CS<2:0> |         |       | —     | WINDOW<2:0> |              |       |                     | <a href="#">173</a> |
| WDTPSL  | PSCNT<7:0>  |         |         |       |       |             |              |       | <a href="#">174</a> |                     |
| WDTPSH  | PSCNT<15:8> |         |         |       |       |             |              |       | <a href="#">174</a> |                     |
| WDTTMR  | WDTTMR<4:0> |         |         |       |       | STATE       | PSCNT<17:16> |       | <a href="#">175</a> |                     |

**Legend:** x = unknown, u = unchanged, – = unimplemented locations read as '0'. Shaded cells are not used by Windowed Watchdog Timer.



## 12.0 8x8 HARDWARE MULTIPLIER

### 12.1 Introduction

All PIC18 devices include an 8x8 hardware multiplier as part of the ALU. The multiplier performs an unsigned operation and yields a 16-bit result that is stored in the product register pair, PRODH:PRODL. The multiplier's operation does not affect any flags in the STATUS register.

Making multiplication a hardware operation allows it to be completed in a single instruction cycle. This has the advantages of higher computational throughput and reduced code size for multiplication algorithms and allows the PIC18 devices to be used in many applications previously reserved for digital signal processors. A comparison of various hardware and software multiply operations, along with the savings in memory and execution time, is shown in [Table 12-1](#).

### 12.2 Operation

[Example 12-1](#) shows the instruction sequence for an 8x8 unsigned multiplication. Only one instruction is required when one of the arguments is already loaded in the WREG register.

[Example 12-2](#) shows the sequence to do an 8x8 signed multiplication. To account for the sign bits of the arguments, each argument's Most Significant bit (MSb) is tested and the appropriate subtractions are done.

#### EXAMPLE 12-1: 8x8 UNSIGNED MULTIPLY ROUTINE

```
MOVF  ARG1, W    ;
MULWF ARG2       ; ARG1 * ARG2 ->
                    ; PRODH:PRODL
```

#### EXAMPLE 12-2: 8x8 SIGNED MULTIPLY ROUTINE

```
MOVF  ARG1, W    ;
MULWF ARG2       ; ARG1 * ARG2 ->
                    ; PRODH:PRODL
BTFSC ARG2, SB   ; Test Sign Bit
SUBWF PRODH, F   ; PRODH = PRODH
                    ;          - ARG1

MOVF  ARG2, W    ;
BTFSC ARG1, SB   ; Test Sign Bit
SUBWF PRODH, F   ; PRODH = PRODH
                    ;          - ARG2
```

**TABLE 12-1: PERFORMANCE COMPARISON FOR VARIOUS MULTIPLY OPERATIONS**

| Routine        | Multiply Method           | Program Memory (Words) | Cycles (Max) | Time         |              |               |             |
|----------------|---------------------------|------------------------|--------------|--------------|--------------|---------------|-------------|
|                |                           |                        |              | @ 64 MHz     | @ 40 MHz     | @ 10 MHz      | @ 4 MHz     |
| 8x8 unsigned   | Without hardware multiply | 13                     | 69           | 4.3 $\mu$ s  | 6.9 $\mu$ s  | 27.6 $\mu$ s  | 69 $\mu$ s  |
|                | Hardware multiply         | 1                      | 1            | 62.5 ns      | 100 ns       | 400 ns        | 1 $\mu$ s   |
| 8x8 signed     | Without hardware multiply | 33                     | 91           | 5.7 $\mu$ s  | 9.1 $\mu$ s  | 36.4 $\mu$ s  | 91 $\mu$ s  |
|                | Hardware multiply         | 6                      | 6            | 375 ns       | 600 ns       | 2.4 $\mu$ s   | 6 $\mu$ s   |
| 16x16 unsigned | Without hardware multiply | 21                     | 242          | 15.1 $\mu$ s | 24.2 $\mu$ s | 96.8 $\mu$ s  | 242 $\mu$ s |
|                | Hardware multiply         | 28                     | 28           | 1.8 $\mu$ s  | 2.8 $\mu$ s  | 11.2 $\mu$ s  | 28 $\mu$ s  |
| 16x16 signed   | Without hardware multiply | 52                     | 254          | 15.9 $\mu$ s | 25.4 $\mu$ s | 102.6 $\mu$ s | 254 $\mu$ s |
|                | Hardware multiply         | 35                     | 40           | 2.5 $\mu$ s  | 4.0 $\mu$ s  | 16.0 $\mu$ s  | 40 $\mu$ s  |

**Example 12-3** shows the sequence to do a 16 x 16 unsigned multiplication. **Equation 12-1** shows the algorithm that is used. The 32-bit result is stored in four registers (RES<3:0>).

## EQUATION 12-1: 16 x 16 UNSIGNED MULTIPLICATION ALGORITHM

$$\begin{aligned} \text{RES3:RES0} &= \text{ARG1H:ARG1L} \bullet \text{ARG2H:ARG2L} \\ &= (\text{ARG1H} \bullet \text{ARG2H} \bullet 2^{16}) + \\ &\quad (\text{ARG1H} \bullet \text{ARG2L} \bullet 2^8) + \\ &\quad (\text{ARG1L} \bullet \text{ARG2H} \bullet 2^8) + \\ &\quad (\text{ARG1L} \bullet \text{ARG2L}) \end{aligned}$$

## EXAMPLE 12-3: 16 x 16 UNSIGNED MULTIPLY ROUTINE

```

MOVF ARG1L, W
MULWF ARG2L           ; ARG1L * ARG2L->
                        ; PRODH:PRODL

MOVFF PRODH, RES1
MOVFF PRODL, RES0
;

MOVF ARG1H, W
MULWF ARG2H           ; ARG1H * ARG2H->
                        ; PRODH:PRODL

MOVFF PRODH, RES3
MOVFF PRODL, RES2
;

MOVF ARG1L, W
MULWF ARG2H           ; ARG1L * ARG2H->
                        ; PRODH:PRODL

MOVF PRODL, W
ADDWF RES1, F         ; Add cross
MOVF PRODH, W         ; products
ADDWFC RES2, F
CLRF WREG
ADDWFC RES3, F
;

MOVF ARG1H, W
MULWF ARG2L           ; ARG1H * ARG2L->
                        ; PRODH:PRODL

MOVF PRODL, W
ADDWF RES1, F         ; Add cross
MOVF PRODH, W         ; products
ADDWFC RES2, F
CLRF WREG
ADDWFC RES3, F
;

```

**Example 12-4** shows the sequence to do a 16 x 16 signed multiply. **Equation 12-2** shows the algorithm used. The 32-bit result is stored in four registers (RES<3:0>). To account for the sign bits of the arguments, the MSb for each argument pair is tested and the appropriate subtractions are done.

## EQUATION 12-2: 16 x 16 SIGNED MULTIPLICATION ALGORITHM

$$\begin{aligned} \text{RES3:RES0} &= \text{ARG1H:ARG1L} \bullet \text{ARG2H:ARG2L} \\ &= (\text{ARG1H} \bullet \text{ARG2H} \bullet 2^{16}) + \\ &\quad (\text{ARG1H} \bullet \text{ARG2L} \bullet 2^8) + \\ &\quad (\text{ARG1L} \bullet \text{ARG2H} \bullet 2^8) + \\ &\quad (\text{ARG1L} \bullet \text{ARG2L}) + \\ &\quad (-1 \bullet \text{ARG2H} <7> \bullet \text{ARG1H:ARG1L} \bullet 2^{16}) + \\ &\quad (-1 \bullet \text{ARG1H} <7> \bullet \text{ARG2H:ARG2L} \bullet 2^{16}) \end{aligned}$$

## EXAMPLE 12-4: 16 x 16 SIGNED MULTIPLY ROUTINE

```

MOVF ARG1L, W
MULWF ARG2L           ; ARG1L * ARG2L ->
                        ; PRODH:PRODL

MOVFF PRODH, RES1
MOVFF PRODL, RES0
;

MOVF ARG1H, W
MULWF ARG2H           ; ARG1H * ARG2H ->
                        ; PRODH:PRODL

MOVFF PRODH, RES3
MOVFF PRODL, RES2
;

MOVF ARG1L, W
MULWF ARG2H           ; ARG1L * ARG2H ->
                        ; PRODH:PRODL

MOVF PRODL, W
ADDWF RES1, F         ; Add cross
MOVF PRODH, W         ; products
ADDWFC RES2, F
CLRF WREG
ADDWFC RES3, F
;

MOVF ARG1H, W
MULWF ARG2L           ; ARG1H * ARG2L ->
                        ; PRODH:PRODL

MOVF PRODL, W
ADDWF RES1, F         ; Add cross
MOVF PRODH, W         ; products
ADDWFC RES2, F
CLRF WREG
ADDWFC RES3, F
;

BTFSS ARG2H, 7        ; ARG2H:ARG2L neg?
BRA SIGN_ARG1         ; no, check ARG1
MOVF ARG1L, W
SUBWF RES2
MOVF ARG1H, W
SUBWFB RES3
;

SIGN_ARG1
BTFSS ARG1H, 7        ; ARG1H:ARG1L neg?
BRA CONT_CODE         ; no, done
MOVF ARG2L, W
SUBWF RES2
MOVF ARG2H, W
SUBWFB RES3
;

CONT_CODE
:

```

## 13.0 NONVOLATILE MEMORY (NVM) CONTROL

Nonvolatile Memory (NVM) is separated into two types: Program Flash Memory (PFM) and Data EEPROM Memory.

PFM, Data EEPROM, User IDs and Configuration bits can all be accessed using the REG<1:0> bits of the NVMCON1 register.

The write time is controlled by an on-chip timer. The write/erase voltages are generated by an on-chip charge pump rated to operate over the operating voltage range of the device.

NVM can be protected in two ways, by either code protection or write protection. Code protection ( $\overline{CP}$  and  $\overline{CPD}$  bits in Configuration Word 5L) disables access, reading and writing to both PFM and Data EEPROM Memory via external device programmers. Code protection does not affect the self-write and erase functionality. Code protection can only be reset by a device programmer performing a Bulk Erase to the device, clearing all nonvolatile memory, Configuration bits and User IDs.

Write protection prohibits self-write and erase to a portion or all of the PFM, as defined by the WRT bits of Configuration Word 4H. Write protection does not affect a device programmer's ability to read, write or erase the device. 2017-2020

**TABLE 13-1: NVM ORGANIZATION AND ACCESS INFORMATION**

| Memory                                 | PC<20:0><br>ICSP™ Addr<21:0><br>TBLPTR<21:0><br>NVMADDR<9:0> | Execution     | User Access      |                               |                               |
|----------------------------------------|--------------------------------------------------------------|---------------|------------------|-------------------------------|-------------------------------|
|                                        |                                                              | CPU Execution | REG              | TABLAT                        | NVMDAT                        |
| Program Flash Memory (PFM)             | 00 0000h<br>...<br>01 FFFFh                                  | Read          | 10               | Read/<br>Write <sup>(1)</sup> | — <sup>(3)</sup>              |
| User IDs <sup>(2)</sup>                | 20 0000h<br>...<br>20 000Fh                                  | No Access     | x1               | Read/<br>Write                | — <sup>(3)</sup>              |
| Reserved                               | 20 0010h<br>...<br>2F FFFFh                                  | No Access     | — <sup>(3)</sup> |                               |                               |
| Configuration                          | 30 0000h<br>...<br>30 0009h                                  | No Access     | x1               | Read/<br>Write <sup>(1)</sup> | — <sup>(3)</sup>              |
| Reserved                               | 30 000Ah<br>...<br>30 FFFFh                                  | No Access     | — <sup>(3)</sup> |                               |                               |
| User Data Memory (Data EEPROM)         | 31 0000h<br>...<br>31 03FFh                                  | No Access     | 00               | — <sup>(3)</sup>              | Read/<br>Write <sup>(1)</sup> |
| Reserved                               | 31 0400h<br>...<br>3E FFFFh                                  | No Access     | — <sup>(3)</sup> |                               |                               |
| Device Information Area (DIA)          | 3F 0000h<br>...<br>3F 003Fh                                  | No Access     | x1               | Read                          | — <sup>(3)</sup>              |
| Reserved                               | 3F 0040h<br>...<br>3F FF09h                                  | No Access     | — <sup>(3)</sup> |                               |                               |
| Device Configuration Information (DCI) | 3F FF00h<br>...<br>3F FF09h                                  | No Access     | x1               | Read                          | — <sup>(3)</sup>              |
| Reserved                               | 3F FF0Ah<br>...<br>3F FFFBh                                  | No Access     | — <sup>(3)</sup> |                               |                               |
| Revision ID/<br>Device ID              | 3F FFFCh<br>...<br>3F FFFFh                                  | No Access     | x1               | Read                          | — <sup>(3)</sup>              |

**Note 1:** Subject to Memory Write Protection settings.

**Note 2:** User IDs are eight words ONLY. There is no code protection, table read protection or write protection implemented for this region.

**Note 3:** Reads as '0', writes clear the WR bit and WRERR bit is set.

## 13.1 Program Flash Memory

The Program Flash Memory is readable, writable and erasable during normal operation over the entire VDD range.

A read from program memory is executed one byte at a time. A write to program memory or program memory erase is executed on blocks of  $n$  bytes at a time. Refer to [Table 5-4](#) for write and erase block sizes. A Bulk Erase operation cannot be issued from user code.

Writing or erasing program memory will cease instruction fetches until the operation is complete. The program memory cannot be accessed during the write or erase, therefore, code cannot execute. An internal programming timer terminates program memory writes and erases.

A value written to program memory does not need to be a valid instruction. Executing a program memory location that forms an invalid instruction results in a NOP.

It is important to understand the PFM memory structure for erase and programming operations. Program memory word size is 16 bits wide. PFM is arranged in

rows. A row is the minimum size that can be erased by user software. Refer to [Table 5-4](#) for the row sizes for the these devices.

After a row has been erased, all or a portion of this row can be programmed. Data to be written into the program memory row is written to 8-bit wide data write latches by means of 6 address lines. These latches are not directly accessible, but may be loaded via sequential writes to the TABLAT register.

**Note:** To modify only a portion of a previously programmed row, then the contents of the entire row must be read and saved in RAM prior to the erase. Then, the new data and retained data can be written into the write latches to reprogram the row of PFM. However, any unprogrammed locations can be written without first erasing the row. In this case, it is not necessary to save and rewrite the other previously programmed locations

**TABLE 13-2: FLASH MEMORY ORGANIZATION BY DEVICE**

| Device         | Row Erase Size (Words) | Write Latches (Bytes) | Program Flash Memory (Words) | Data Memory (Bytes) |
|----------------|------------------------|-----------------------|------------------------------|---------------------|
| PIC18(L)F25K83 | 64                     | 128                   | 16384                        | 1024                |
| PIC18(L)F26K83 | 64                     | 128                   | 32768                        | 1024                |

## 13.1.1 TABLE READS AND TABLE WRITES

In order to read and write program memory, there are two operations that allow the processor to move bytes between the program memory space and the data RAM:

- Table Read (TBLRD)
- Table Write (TBLWT)

The program memory space is 16 bits wide, while the data RAM space is eight bits wide. Table reads and table writes move data between these two memory spaces through an 8-bit register (TABLAT).

The table read operation retrieves one byte of data directly from program memory and places it into the TABLAT register. Figure 13-1 shows the operation of a table read.

The table write operation stores one byte of data from the TABLAT register into a write block holding register. The procedure to write the contents of the holding registers into program memory is detailed in [Section 13.1.6 “Writing to Program Flash Memory”](#). Figure 13-2 shows the operation of a table write with program memory and data RAM.

Table operations work with byte entities. Tables containing data, rather than program instructions, are not required to be word aligned. Therefore, a table can start and end at any byte address. If a table write is being used to write executable code into program memory, program instructions will need to be word aligned.

**FIGURE 13-1: TABLE READ OPERATION**

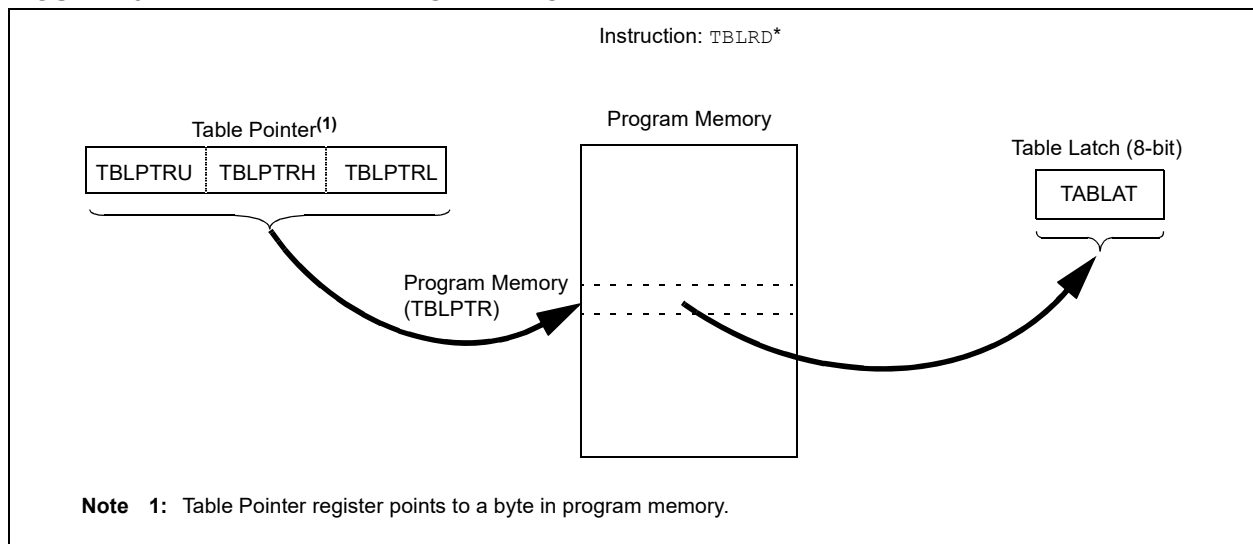
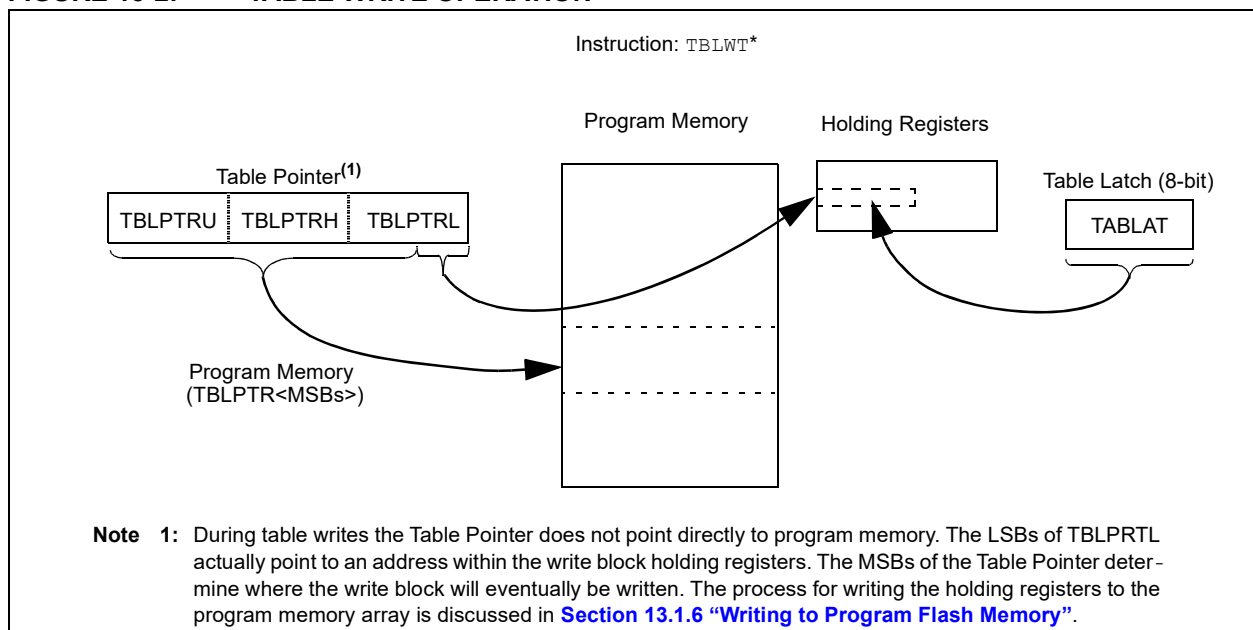


**FIGURE 13-2: TABLE WRITE OPERATION**



## 13.1.2 CONTROL REGISTERS

Several control registers are used in conjunction with the `TBLRD` and `TBLWT` instructions. These include the following registers:

- NVMCON1 register
- NVMCON2 register
- TABLAT register
- TBLPTR registers

### 13.1.2.1 NVMCON1 and NVMCON2 Registers

The NVMCON1 register ([Register 13-1](#)) is the control register for memory accesses. The NVMCON2 register is not a physical register; it is used exclusively in the memory write and erase sequences. Reading NVMCON2 will read all '0's.

The `REG<1:0>` control bits determine if the access will be to Data EEPROM Memory locations, PFM locations or User IDs, Configuration bits, Rev ID and Device ID. When `REG<1:0> = 00`, any subsequent operations will operate on the Data EEPROM Memory. When `REG<1:0> = 10`, any subsequent operations will operate on the program memory. When `REG<1:0> = x1`, any subsequent operations will operate on the Configuration bits, User IDs, Rev ID and Device ID.

The `FREE` bit allows the program memory erase operation. When the `FREE` bit is set, an erase operation is initiated on the next `WR` command. When `FREE` is clear, only writes are enabled. This bit is applicable only to the PFM and not to data EEPROM.

When set, the `WREN` bit will allow a program/erase operation. The `WREN` bit is cleared on power-up.

The `WRERR` bit is set by hardware when the `WR` bit is set and is cleared when the internal programming timer expires and the write operation is successfully complete.

The `WR` control bit initiates erase/write cycle operation when the `REG<1:0>` bits point to the Data EEPROM Memory location, and it initiates a write operation when the `REG<1:0>` bits point to the PFM location. The `WR` bit cannot be cleared by firmware; it can only be set by firmware. Then the `WR` bit is cleared by hardware at the completion of the write operation.

The `NVMIF` Interrupt Flag bit is set when the write is complete. The `NVMIF` flag stays set until cleared by firmware.

### 13.1.2.2 TABLAT – Table Latch Register

The Table Latch (`TABLAT`) is an 8-bit register mapped into the SFR space. The Table Latch register is used to hold 8-bit data during data transfers between program memory and data RAM.

### 13.1.2.3 TBLPTR – Table Pointer Register

The Table Pointer (`TBLPTR`) register addresses a byte within the program memory. The `TBLPTR` is comprised of three SFR registers: Table Pointer Upper Byte, Table Pointer High Byte and Table Pointer Low Byte (`TBLPTRU:TBLPTRH:TBLPTRL`). These three registers join to form a 22-bit wide pointer. The low-order 21 bits allow the device to address up to 2 Mbytes of program memory space. The 22<sup>nd</sup> bit allows access to the Device ID, the User ID and the Configuration bits.

The Table Pointer register, `TBLPTR`, is used by the `TBLRD` and `TBLWT` instructions. These instructions can update the `TBLPTR` in one of four ways based on the table operation. These operations on the `TBLPTR` affect only the low-order 21 bits.

### 13.1.2.4 Table Pointer Boundaries

`TBLPTR` is used in reads, writes and erases of the Program Flash Memory.

When a `TBLRD` is executed, all 22 bits of the `TBLPTR` determine which byte is read from program memory directly into the `TABLAT` register.

When a `TBLWT` is executed the byte in the `TABLAT` register is written, not to memory but, to a holding register in preparation for a program memory write. The holding registers constitute a write block which varies depending on the device (see [Table 5-4](#)). The 6 LSBs of the `TBLPTRL` register determine which specific address within the holding register block is written to. The MSBs of the Table Pointer have no effect during `TBLWT` operations.

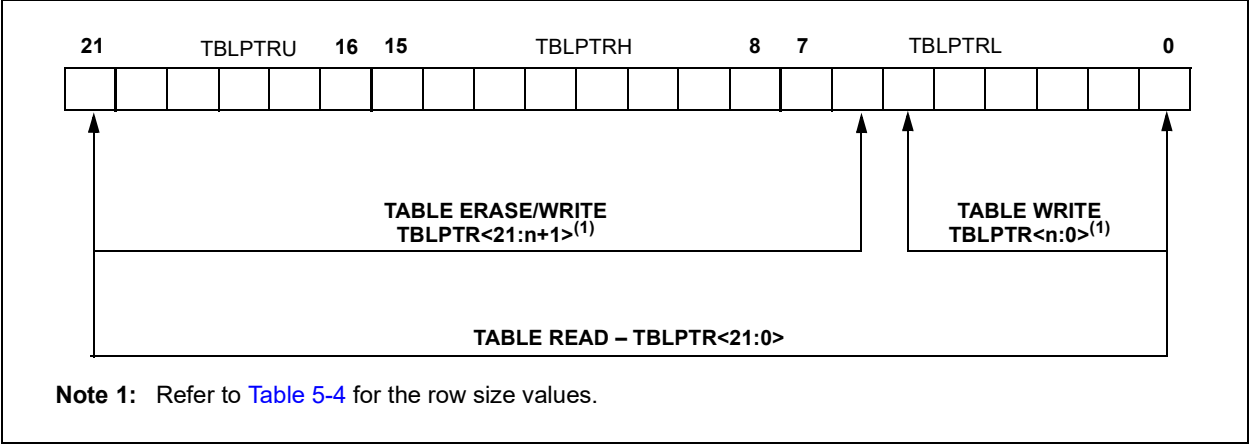
When a program memory write is executed the entire holding register block is written to the memory at the address determined by the MSBs of the `TBLPTR`. The 6 LSBs are ignored during memory writes. For more detail, see [Section 13.1.6 “Writing to Program Flash Memory”](#).

[Figure 13-3](#) describes the relevant boundaries of `TBLPTR` based on Program Flash Memory operations.

TABLE 13-3: TABLE POINTER OPERATIONS WITH TBLRD AND TBLWT INSTRUCTIONS

| Example            | Operation on Table Pointer                  |
|--------------------|---------------------------------------------|
| TBLRD*<br>TBLWT*   | TBLPTR is not modified                      |
| TBLRD*+<br>TBLWT*+ | TBLPTR is incremented after the read/write  |
| TBLRD*-<br>TBLWT*- | TBLPTR is decremented after the read/write  |
| TBLRD++<br>TBLWT++ | TBLPTR is incremented before the read/write |

FIGURE 13-3: TABLE POINTER BOUNDARIES BASED ON OPERATION



## 13.1.3 READING THE PROGRAM FLASH MEMORY

The **TBLRD** instruction retrieves data from program memory and places it into data RAM. Table reads from program memory are performed one byte at a time.

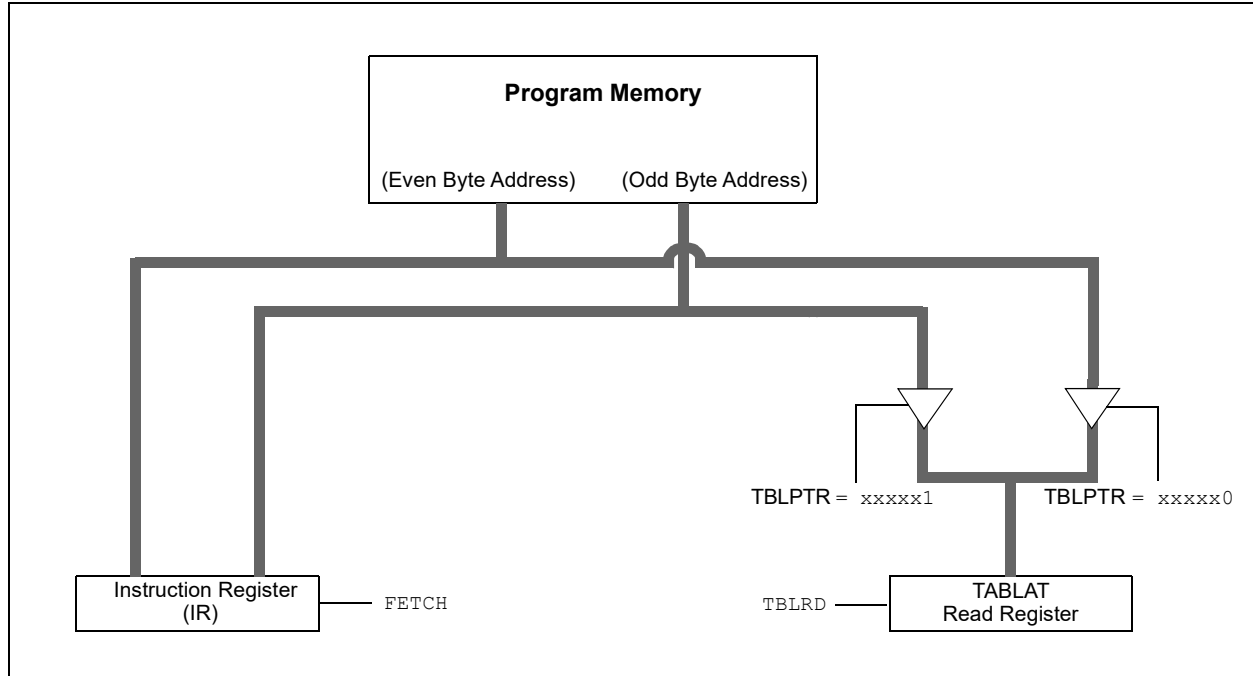
**TBLPTR** points to a byte address in program space. Executing **TBLRD** places the byte pointed to into **TABLAT**. In addition, **TBLPTR** can be modified automatically for the next table read operation.

The CPU operation is suspended during the read, and it resumes immediately after. From the user point of view, **TABLAT** is valid in the next instruction cycle.

The internal program memory is typically organized by words. The Least Significant bit of the address selects between the high and low bytes of the word.

Figure 13-4 shows the interface between the internal program memory and the **TABLAT**.

**FIGURE 13-4: READS FROM PROGRAM FLASH MEMORY**



**EXAMPLE 13-1: READING A PROGRAM FLASH MEMORY WORD**

```

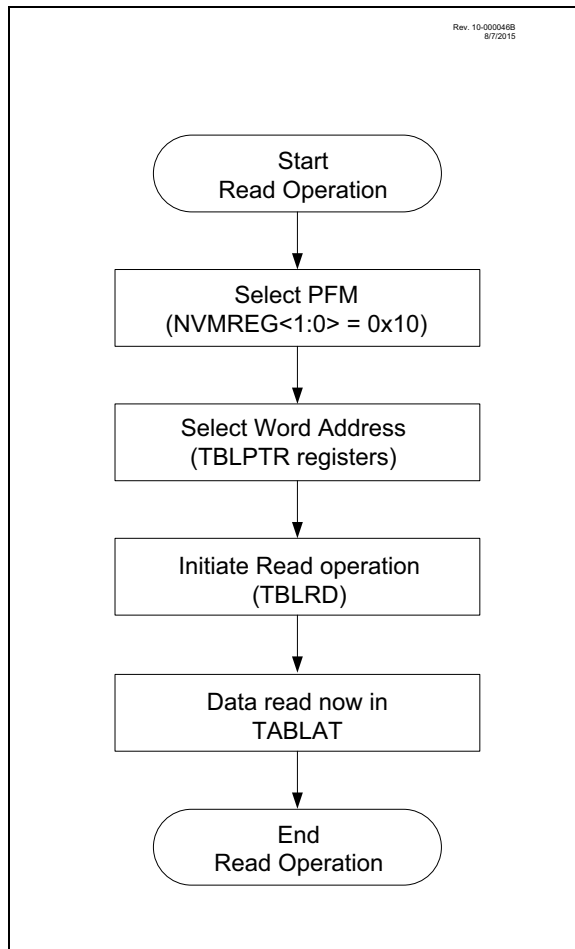
BCF      NVMCON1, REG0      ; point to Program Flash Memory
BSF      NVMCON1, REG1      ; access Program Flash Memory
MOVLW    CODE_ADDR_UPPER    ; Load TBLPTR with the base
MOVWF    TBLPTRU             ; address of the word
MOVLW    CODE_ADDR_HIGH
MOVWF    TBLPTRH
MOVLW    CODE_ADDR_LOW
MOVWF    TBLPTRL

READ_WORD
TBLRD*+      ; read into TABLAT and increment
MOV     TABLAT, W           ; get data
MOVWF    WORD_EVEN

TBLRD*+      ; read into TABLAT and increment
MOV     TABLAT, W           ; get data
MOV     WORD_ODD
    
```



**FIGURE 13-5: PROGRAM FLASH  
MEMORY READ  
FLOWCHART**



## 13.1.4 NVM UNLOCK SEQUENCE

The unlock sequence is a mechanism that protects the NVM from unintended self-write programming or erasing. The sequence must be executed and completed without interruption to successfully complete any of the following operations:

- PFM Row Erase
- Write of PFM write latches to PFM memory
- Write of PFM write latches to User IDs
- Write to Data EEPROM Memory
- Write to Configuration Words

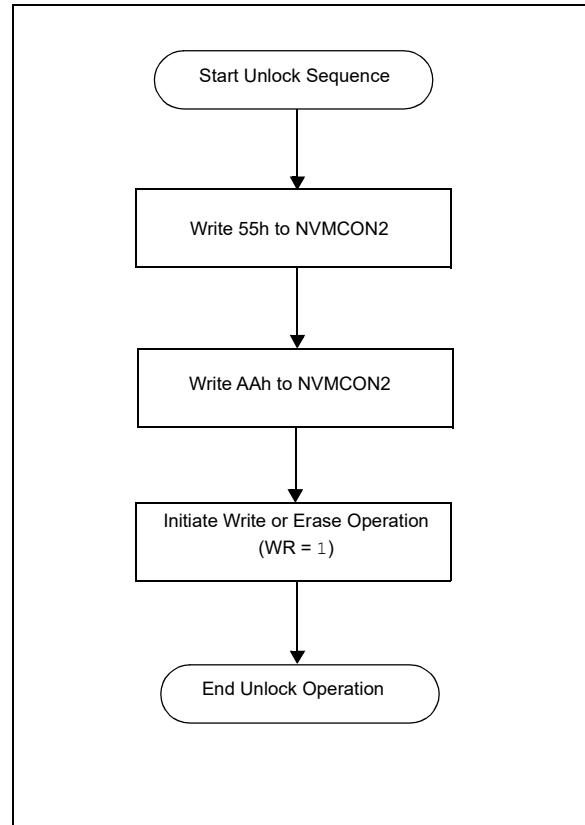
The unlock sequence consists of the following steps and must be completed in order:

- Write 55h to NVMCON2
- Write AAh to NMVCON2
- Set the WR bit of NVMCON1

Once the WR bit is set, the processor will stall internal operations until the operation is complete and then resume with the next instruction.

Since the unlock sequence must not be interrupted, global interrupts should be disabled prior to the unlock sequence and re-enabled after the unlock sequence is completed.

**FIGURE 13-6: NVM UNLOCK SEQUENCE FLOWCHART**



### EXAMPLE 13-2: NVM UNLOCK SEQUENCE

|         |              |                                              |
|---------|--------------|----------------------------------------------|
| BCF     | INTCON0,GIE  | ; Recommended so sequence is not interrupted |
| BANKSEL | NVMCON1      |                                              |
| BSF     | NVMCON1,WREN | ; Enable write/erase                         |
| MOVLW   | 55h          | ; Load 55h                                   |
|         |              |                                              |
| MOVWF   | NVMCON2      | ; Step 1: Load 55h into NVMCON2              |
| MOVLW   | AAh          | ; Step 2: Load W with AAh                    |
| MOVWF   | NVMCON2      | ; Step 3: Load AAh into NVMCON2              |
| BSF     | INTCON1,WR   | ; Step 4: Set WR bit to begin write/erase    |
|         |              |                                              |
| BSF     | INTCON0,GIE  | ; Re-enable interrupts                       |

**Note 1:** Sequence begins when NVMCON2 is written; steps 1-4 must occur in the cycle-accurate order shown. If the timing of the steps 1 to 4 is corrupted by an interrupt or a debugger Halt, the action will not take place.

**2:** Opcodes shown are illustrative; any instruction that has the indicated effect may be used.

## 13.1.5 ERASING PROGRAM FLASH MEMORY

The minimum erase block is 64 words (refer to [Table 5-4](#)). Only through the use of an external programmer, or through ICSP™ control, can larger blocks of program memory be bulk erased. Word erase in the program memory array is not supported.

For example, when initiating an erase sequence from a microcontroller with erase row size of 64 words, a block of 64 words (128 bytes) of program memory is erased. The Most Significant 16 bits of the TBLPTR<21:6> point to the block being erased. The TBLPTR<5:0> bits are ignored.

The NVMCON1 register commands the erase operation. The REG<1:0> bits must be set to point to the Program Flash Memory. The WREN bit must be set to enable write operations. The FREE bit is set to select an erase operation.

The NVM unlock sequence described in [Section 13.1.4 “NVM Unlock Sequence”](#) should be used to guard against accidental writes. This is sometimes referred to as a long write.

A long write is necessary for erasing program memory. Instruction execution is halted during the long write cycle. The long write is terminated by the internal programming timer.

### 13.1.5.1 Program Flash Memory Erase Sequence

The sequence of events for erasing a block of internal program memory is:

1. REG bits of the NVMCON1 register point to PFM
2. Set the FREE and WREN bits of the NVMCON1 register
3. Perform the unlock sequence as described in [Section 13.1.4 “NVM Unlock Sequence”](#)

If the PFM address is write-protected, the WR bit will be cleared and the erase operation will not take place, WRERR is signaled in this scenario.

The operation erases the memory row indicated by masking the LSBs of the current TBLPTR.

While erasing PFM, CPU operation is suspended and it resumes when the operation is complete. Upon completion the WR bit is cleared in hardware, the NVMIF is set and an interrupt will occur if the NVMIE bit is also set.

Write latch data is not affected by erase operations and WREN will remain unchanged.

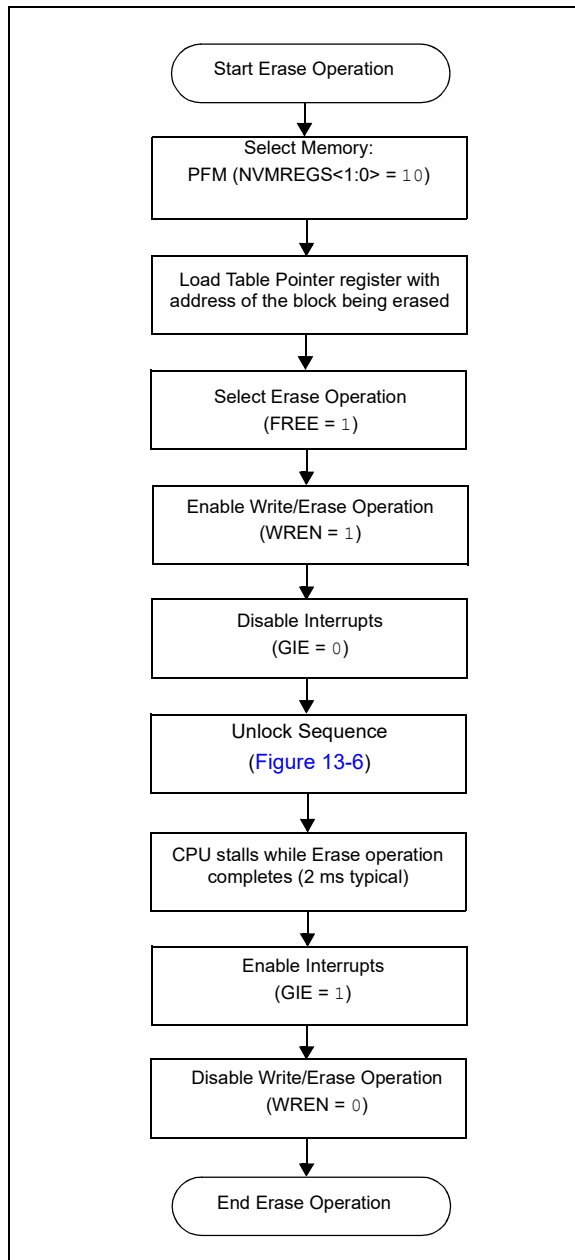
- Note 1:** If a write or erase operation is terminated by an unexpected event, WRERR bit will be set which the user can check to decide whether a rewrite of the location(s) is needed.
- 2: WRERR is set if WR is written to ‘1’ while TBLPTR points to a write-protected address.
  - 3: WRERR is set if WR is written to ‘1’ while TBLPTR points to an invalid address location ([Table 13-1](#)).

## EXAMPLE 13-3: ERASING A PROGRAM FLASH MEMORY BLOCK

; This sample row erase routine assumes the following:  
; 1. A valid address within the erase row is loaded in variables TBLPTR register

```
        CLRF      NVMCON1      ; Setup PFM Access
        MOVLW     CODE_ADDR_UPPER ; load TBLPTR with the base
        MOVWF     TBLPTRU      ; address of the memory block
        MOVLW     CODE_ADDR_HIGH
        MOVWF     TBLPTRH
        MOVLW     CODE_ADDR_LOW
        MOVWF     TBLPTRL
ERASE_BLOCK
        BCF       NVMCON1, REG0 ; point to Program Flash Memory
        BSF       NVMCON1, REG1 ; access Program Flash Memory
        BSF       NVMCON1, WREN  ; enable write to memory
        BSF       NVMCON1, FREE  ; enable block Erase operation
        BCF       INTCON0, GIE   ; disable interrupts
        MOVLW     55h
Required MOVWF     NVMCON2      ; write 55h
Sequence MOVLW     AAh
        MOVWF     NVMCON2      ; write AAh
        BSF       NVMCON1, WR    ; start erase (CPU stalls)
        BSF       INTCON0, GIE   ; re-enable interrupts
```

**FIGURE 13-7: PFM ROW ERASE FLOWCHART**



## 13.1.6 WRITING TO PROGRAM FLASH MEMORY

The programming write block size is described in [Table 5-4](#). Word or byte programming is not supported. Table writes are used internally to load the holding registers needed to program the memory. There are only as many holding registers as there are bytes in a write block. Refer to [Table 5-4](#) for write latch size.

Since the table latch (TABLAT) is only a single byte, the TBLWT instruction needs to be executed multiple times for each programming operation. The write protection state is ignored for this operation. All of the table write operations will essentially be short writes because only the holding registers are written. NVMIF is not affected while writing to the holding registers.

After all the holding registers have been written, the programming operation of that block of memory is started by configuring the NVMCON1 register for a program memory write and performing the long write sequence.

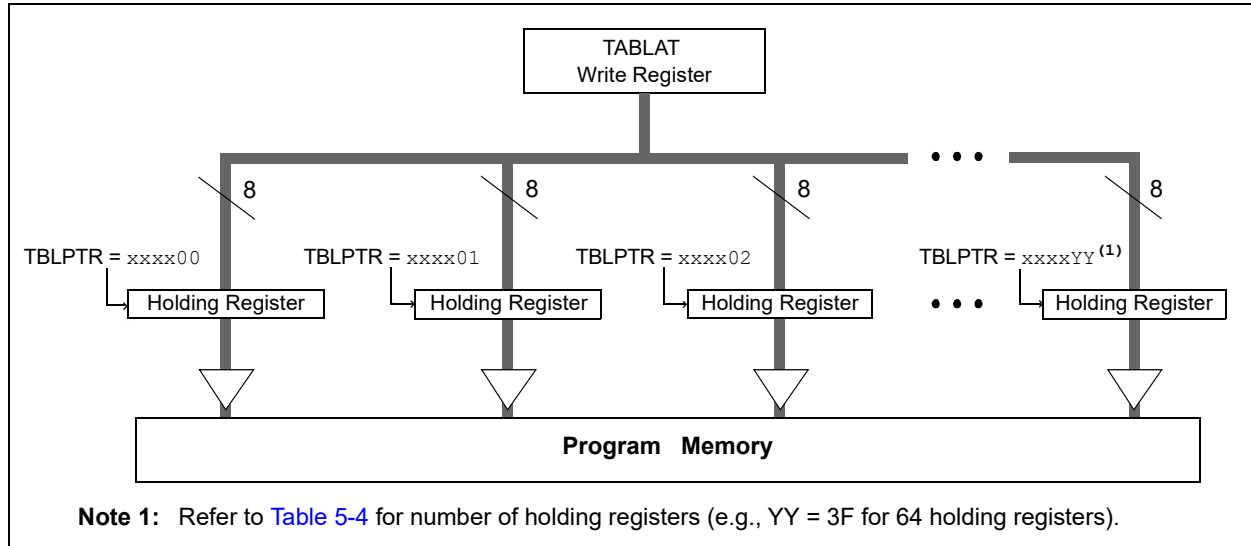
If the PFM address in the TBLPTR is write-protected or if TBLPTR points to an invalid location, the WR bit is cleared without any effect and the WRERR is signaled.

The long write is necessary for programming the program memory. CPU operation is suspended during a long write cycle and resumes when the operation is complete. The long write operation completes in one instruction cycle. When complete, WR is cleared in hardware and NVMIF is set and an interrupt will occur if NVMIE is also set. The latched data is reset to all '1's'. WREN is not changed.

The internal programming timer controls the write time. The write/erase voltages are generated by an on-chip charge pump, rated to operate over the voltage range of the device.

**Note:** The default value of the holding registers on device Resets and after write operations is FFh. A write of FFh to a holding register does not modify that byte. This means that individual bytes of program memory may be modified, provided that the change does not attempt to change any bit from a '0' to a '1'. When modifying individual bytes, it is not necessary to load all holding registers before executing a long write operation.

**FIGURE 13-8: TABLE WRITES TO PROGRAM FLASH MEMORY**



## 13.1.6.1 Program Flash Memory Write Sequence

The sequence of events for programming an internal program memory location should be:

1. Read appropriate number of bytes into RAM. Refer to [Table 5-4](#) for Write latch size.
2. Update data values in RAM as necessary.
3. Load Table Pointer register with address being erased.
4. Execute the block erase procedure.
5. Load Table Pointer register with address of first byte being written.
6. Write the n-byte block into the holding registers with auto-increment. Refer to [Table 5-4](#) for Write latch size.
7. Set REG<1:0> bits to point to program memory.
8. Clear FREE bit and set WREN bit in NVMCON1 register.
9. Disable interrupts.
10. Execute the unlock sequence (see [Section 13.1.4 “NVM Unlock Sequence”](#)).
11. WR bit is set in NVMCON1 register.
12. The CPU will stall for the duration of the write (about 2 ms using internal timer).
13. Re-enable interrupts.
14. Verify the memory (table read).

This procedure will require about 6 ms to update each write block of memory. An example of the required code is given in [Example 13-4](#).

**Note:** Before setting the WR bit, the Table Pointer address needs to be within the intended address range of the bytes in the holding registers.

## EXAMPLE 13-4: WRITING TO PROGRAM FLASH MEMORY

```

        MOVLW    D'64'                ; number of bytes in erase block
        MOVWF    COUNTER
        MOVLW    BUFFER_ADDR_HIGH     ; point to buffer
        MOVWF    FSR0H
        MOVLW    BUFFER_ADDR_LOW
        MOVWF    FSR0L
        MOVLW    CODE_ADDR_UPPER      ; Load TBLPTR with the base
        MOVWF    TBLPTRU               ; address of the memory block
        MOVLW    CODE_ADDR_HIGH
        MOVWF    TBLPTRH
        MOVLW    CODE_ADDR_LOW
        MOVWF    TBLPTRL

READ_BLOCK
        TBLRD*+                        ; read into TABLAT, and inc
        MOVF     TABLAT, W              ; get data
        MOVWF    POSTINC0              ; store data
        DECFSZ   COUNTER                ; done?
        BRA      READ_BLOCK            ; repeat

MODIFY_WORD
        MOVLW    BUFFER_ADDR_HIGH     ; point to buffer
        MOVWF    FSR0H
        MOVLW    BUFFER_ADDR_LOW
        MOVWF    FSR0L
        MOVLW    NEW_DATA_LOW          ; update buffer word
        MOVWF    POSTINC0
        MOVLW    NEW_DATA_HIGH
        MOVWF    INDF0

ERASE_BLOCK
        MOVLW    CODE_ADDR_UPPER      ; load TBLPTR with the base
        MOVWF    TBLPTRU               ; address of the memory block
        MOVLW    CODE_ADDR_HIGH
        MOVWF    TBLPTRH
        MOVLW    CODE_ADDR_LOW
        MOVWF    TBLPTRL
        BCF      NVMCON1, REG0          ; point to Program Flash Memory
        BSF      NVMCON1, REG1          ; point to Program Flash Memory
        BSF      NVMCON1, WREN          ; enable write to memory
        BSF      NVMCON1, FREE          ; enable Erase operation
        BCF      INTCON0, GIE          ; disable interrupts
        MOVLW    55h
        MOVWF    NVMCON2                ; write 55h
        MOVLW    AAh
        MOVWF    NVMCON2                ; write 0AAh
        BSF      NVMCON1, WR            ; start erase (CPU stall)
        BSF      INTCON0, GIE          ; re-enable interrupts
        TBLRD*-                          ; dummy read decrement
        MOVLW    BUFFER_ADDR_HIGH     ; point to buffer
        MOVWF    FSR0H
        MOVLW    BUFFER_ADDR_LOW
        MOVWF    FSR0L

WRITE_BUFFER_BACK
        MOVLW    BlockSize              ; number of bytes in holding register
        MOVWF    COUNTER
        MOVLW    D'64'/BlockSize       ; number of write blocks in 64 bytes
        MOVWF    COUNTER2

```

## EXAMPLE 13-4: WRITING TO PROGRAM FLASH MEMORY (CONTINUED)

```

WRITE_BYTE_TO_HREGS
    MOVF     POSTINC0, W           ; get low byte of buffer data
    MOVWF    TABLAT               ; present data to table latch
    TBLWT+*                       ; write data, perform a short write
                                ; to internal TBLWT holding register.
                                ; loop until holding registers are full

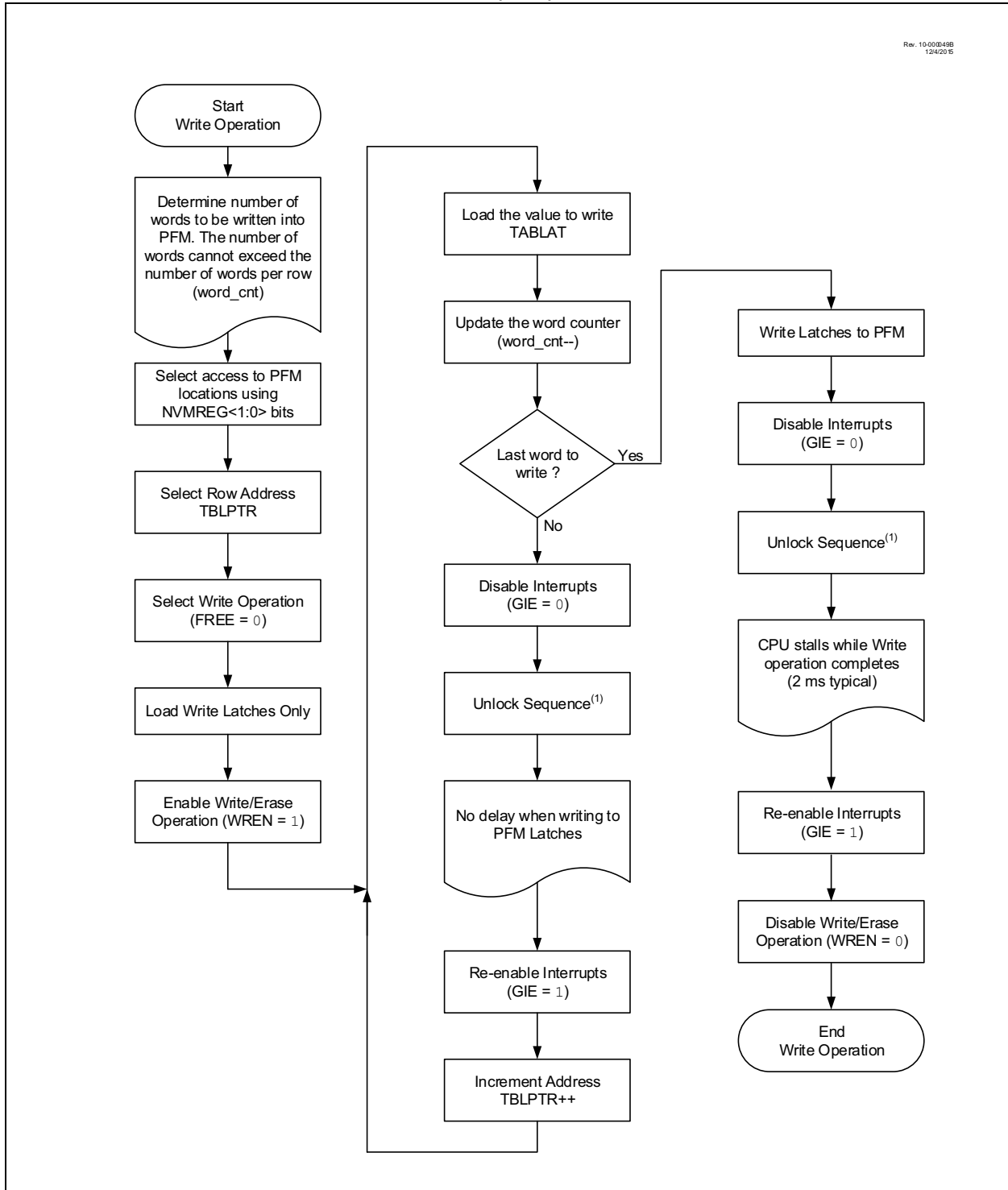
    DECFSZ   COUNTER
    BRA      WRITE_WORD_TO_HREGS

PROGRAM_MEMORY
    BCF      NVMCON1, REG0        ; point to Program Flash Memory
    BSF      NVMCON1, REG1        ; point to Program Flash Memory
    BSF      NVMCON1, WREN        ; enable write to memory
    BCF      NVMCON1, FREE        ; enable write to memory
    BCF      INTCON0, GIE         ; disable interrupts
    MOVLW    55h
Required    MOVWF    NVMCON2        ; write 55h
Sequence    MOVLW    0AAh
    MOVWF    NVMCON2            ; write 0AAh
    BSF      NVMCON1, WR          ; start program (CPU stall)
    DCFSZ    COUNTER2            ; repeat for remaining write blocks
    BRA      WRITE_BYTE_TO_HREGS
    BSF      INTCON0, GIE         ; re-enable interrupts
    BCF      NVMCON1, WREN        ; disable write to memory

```



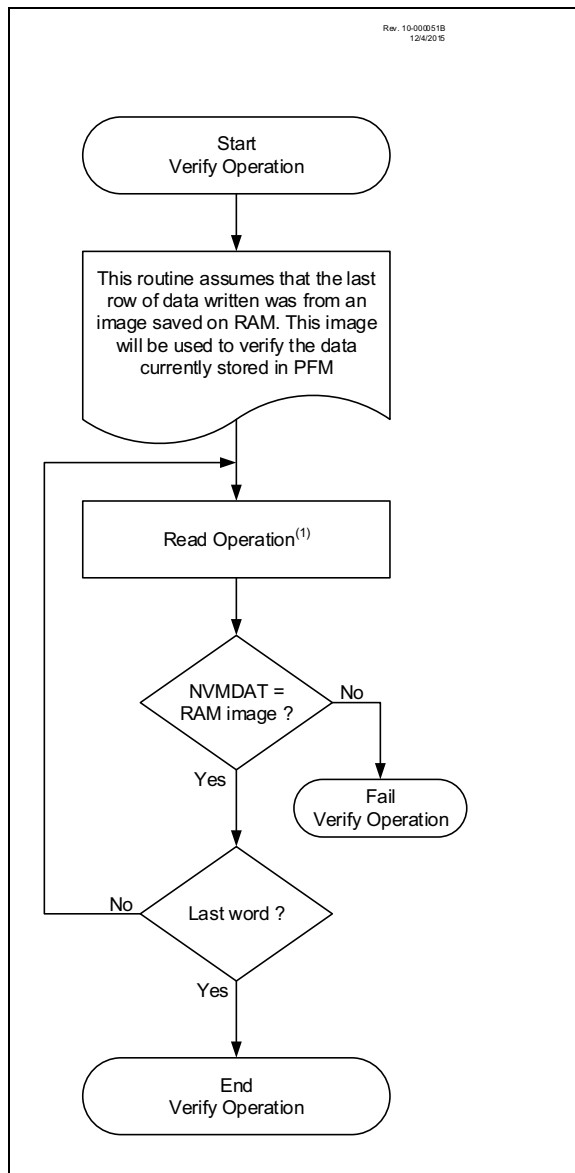
**FIGURE 13-9: PROGRAM FLASH MEMORY (PFM) WRITE FLOWCHART**



## 13.1.6.2 Write Verify

Depending on the application, good programming practice may dictate that the value written to the memory should be verified against the original value. This should be used in applications where excessive writes can stress bits near the specification limit. Since program memory is stored as a full page, the stored program memory contents are compared with the intended data stored in RAM after the last write is complete.

**FIGURE 13-10: PROGRAM FLASH MEMORY VERIFY FLOWCHART**



## 13.1.6.3 Unexpected Termination of Write Operation

If a write is terminated by an unplanned event, such as loss of power or an unexpected Reset, the memory location just programmed should be verified and reprogrammed if needed. If the write operation is interrupted by a MCLR Reset or a WDT Time-out Reset during normal operation, the WRERR bit will be set which the user can check to decide whether a rewrite of the location(s) is needed.

## 13.1.6.4 Protection Against Spurious Writes

A write sequence is valid only when both the following conditions are met, this prevents spurious writes which might lead to data corruption.

1. The WR bit is gated through the WREN bit. It is suggested to have the WREN bit cleared at all times except during memory writes. This prevents memory writes if the WR bit gets set accidentally.
2. The NVM unlock sequence must be performed each time before a write operation.

## 13.2 Device Information Area, Device Configuration Area, User ID, Device ID and Configuration Word Access

When REG<1:0> = 0b01 or 0b11 in the NVMCON1 register, the Device Information Area, the Device Configuration Area, the User ID's, Device ID/Revision ID and Configuration Words can be accessed. Different access may exist for reads and writes (see [Table 13-1](#)).

### 13.2.1 Reading Access

The user can read from these blocks by setting the REG bits to 0b01 or 0b11. The user needs to load the address into the TBLPTR registers. Executing a TBLRD after that moves the byte pointed to the TABLAT register. The CPU operation is suspended during the read and resumes after. When read access is initiated on an address outside the parameters listed in [Table 13-1](#), the TABLAT register is cleared, reading back '0's.

### 13.2.2 Writing Access

The WREN bit in NVMCON1 must be set to enable writes. This prevents accidental writes to the CONFIG words due to errant (unexpected) code execution. The WREN bit should be kept clear at all times, except when updating the CONFIG words. The WREN bit is not cleared by hardware. The WR bit will be inhibited from being set unless the WREN bit is set.

The user needs to load the TBLPTR and TABLAT register with the address and data byte respectively before executing the Write command. An unlock sequence needs to be followed for writing to the USER IDs/DEVICE IDs/CONFIG words ([Section 13.1.4, NVM Unlock Sequence](#)). If WRTC = 0 or if TBLPTR points an invalid address location (see [Table 13-1](#)), WR bit is cleared without any effect and WRERR is set.

A single CONFIG word byte is written at once and the operation includes an implicit erase cycle for that byte (it is not necessary to set FREE). CPU execution is stalled and at the completion of the write cycle, the WR bit is cleared in hardware and the NVM Interrupt Flag bit (NVMIF) is set. The new CONFIG value takes effect when the CPU resumes operation.

**TABLE 13-4: DIA, DCI, USER ID, DEV/REV ID AND CONFIGURATION WORD ACCESS (REG<1:0> = x1)**

| Address           | Function              | Read Access | Write Access |
|-------------------|-----------------------|-------------|--------------|
| 20 0000h-20 000Fh | User IDs              | Yes         | Yes          |
| 30 0000h-30 0009h | Configuration Words   | Yes         | Yes          |
| 3F 0000h-3F 003Fh | DIA                   | Yes         | No           |
| 3F FF00h-3F FF09h | DCI                   | Yes         | No           |
| 3F FFFCh-3F FFFFh | Revision ID/Device ID | Yes         | No           |

## 13.3 Data EEPROM Memory

The data EEPROM is a nonvolatile memory array, separate from the data RAM and program memory, which is used for long-term storage of program data. It is not directly mapped in either the register file or program memory space but is indirectly addressed through the Special Function Registers (SFRs). The EEPROM is readable and writable during normal operation over the entire VDD range.

Four SFRs are used to read and write to the data EEPROM as well as the program memory. They are:

- NVMCON1
- NVMCON2
- NVMDAT
- NVMADRL
- NVMADRH

The data EEPROM allows byte read and write. When interfacing to the data memory block, NVMDAT holds the 8-bit data for read/write and the NVMADRH:NVMADRL register pair holds the address of the EEPROM location being accessed.

The EEPROM data memory is rated for high erase/write cycle endurance. A byte write automatically erases the location and writes the new data (erase-before-write). The write time is controlled by an internal programming timer; it will vary with voltage and temperature as well as from chip-to-chip. Refer to the Data EEPROM Memory parameters in [Section 45.0 “Electrical Specifications”](#) for limits.

### 13.3.1 NVMADRL AND NVMADRH REGISTERS

The NVMADRH:NVMADRL registers are used to address the data EEPROM for read and write operations.

### 13.3.2 NVMCON1 AND NVMCON2 REGISTERS

Access to the data EEPROM is controlled by two registers: NVMCON1 and NVMCON2. These are the same registers which control access to the program memory and are used in a similar manner for the data EEPROM.

The NVMCON1 register ([Register 13-1](#)) is the control register for data and program memory access. Control bits REG<1:0> determine if the access will be to program, Data EEPROM Memory or the User IDs, Configuration bits, Revision ID and Device ID.

The WREN bit, when set, will allow a write operation. On power-up, the WREN bit is clear.

The WRERR bit is set by hardware when the WR bit is set and cleared when the internal programming timer expires and the write operation is complete.

The WR control bit initiates write operations. The bit can be set but not cleared by software. It is cleared only by hardware at the completion of the write operation.

The NVMIF Interrupt Flag bit of the PIR0 register is set when the write is complete. It must be cleared by software.

Control bits, RD and WR, start read and erase/write operations, respectively. These bits are set by firmware and cleared by hardware at the completion of the operation.

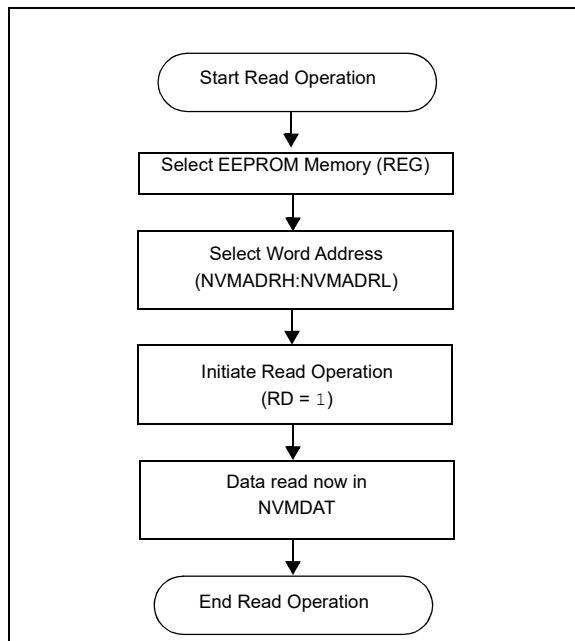
The RD bit cannot be set when accessing program memory (REG<1:0> = 0x10). Program memory is read using table read instructions. See [Section 13.1.1 “Table Reads and Table Writes”](#) regarding table reads.

## 13.3.3 READING THE DATA EEPROM MEMORY

To read a data memory location, the user must write the address to the NVMADRL and NVMADRH register pair, clear REG<1:0> control bit in NVMCON1 register to access Data EEPROM locations and then set control bit, RD. The data is available on the very next instruction cycle; therefore, the NVMDAT register can be read by the next instruction. NVMDAT will hold this value until another read operation, or until it is written to by the user (during a write operation).

The basic process is shown in [Example 13-5](#).

**FIGURE 13-11: DATA EEPROM READ FLOWCHART**



## 13.3.4 WRITING TO THE DATA EEPROM MEMORY

To write an EEPROM data location, the address must first be written to the NVMADRL and NVMADRH register pair and the data written to the NVMDAT register. The sequence in [Example 13-6](#) must be followed to initiate the write cycle.

The write will not begin if NVM Unlock sequence, described in [Section 13.1.4 “NVM Unlock Sequence”](#), is not exactly followed for each byte. It is strongly recommended that interrupts be disabled during this code segment.

Additionally, the WREN bit in NVMCON1 must be set to enable writes. This mechanism prevents accidental writes to data EEPROM due to unexpected code execution (i.e., runaway programs). The WREN bit should be kept clear at all times, except when updating the EEPROM. The WREN bit is not cleared by hardware.

After a write sequence has been initiated, NVMCON1, NVMADRL, NVMADRH and NVMDAT cannot be modified. The WR bit will be inhibited from being set unless the WREN bit is set. Both WR and WREN cannot be set with the same instruction.

After a write sequence has been initiated, clearing the WREN bit will not affect this write cycle. A single Data EEPROM word is written and the operation includes an implicit erase cycle for that word (it is not necessary to set FREE). CPU execution continues in parallel and at the completion of the write cycle, the WR bit is cleared in hardware and the NVM Interrupt Flag bit (NVMIF) is set. The user can either enable this interrupt or poll this bit. NVMIF must be cleared by software.

## 13.3.5 WRITE VERIFY

Depending on the application, good programming practice may dictate that the value written to the memory should be verified against the original value. This should be used in applications where excessive writes can stress bits near the specification limit.

### EXAMPLE 13-5: DATA EEPROM READ

```
; Data Memory Address to read
      CLRF      NVMCON1          ; Setup Data EEPROM Access
      MOVF      EE_ADDRL, W      ;
      MOVWF     NVMADRL          ; Setup Address
      BSF       NVMCON1, RD      ; Issue EE Read
      MOVF      NVMDAT, W        ; W = EE_DATA
```

### EXAMPLE 13-6: DATA EEPROM WRITE

```
; Data Memory Address to write
      CLRF      NVMCON1          ; Setup Data EEPROM Access
      MOVF      EE_ADDRL, W      ;
      MOVWF     NVMADRL          ; Setup Address
; Data Memory Value to write
      MOVF      EE_DATA, W       ;
      MOVWF     NVMDAT           ;
; Enable writes
      BSF       NVMCON1, WREN    ;
; Disable interrupts
      BCF       INTCON0, GIE     ;
; Required unlock sequence
      MOVLW     55h              ;
      MOVWF     NVMCON2          ;
      MOVLW     AAh              ;
      MOVWF     NVMCON2          ;
; Set WR bit to begin write
      BSF       NVMCON1, WR      ;
; Enable INT
      BSF       INTCON0, GIE     ;
; Wait for interrupt, write done
      SLEEP                     ;
; Disable writes
      BCF       NVMCON1, WREN    ;
```

## 13.3.6 OPERATION DURING CODE-PROTECT

Data EEPROM Memory has its own code-protect bits in Configuration Words. External read and write operations are disabled if code protection is enabled.

If the Data EEPROM is write-protected or if NVMADR points an invalid address location, the WR bit is cleared without any effect. WRERR is signaled in this scenario.

## 13.3.7 PROTECTION AGAINST SPURIOUS WRITE

There are conditions when the user may not want to write to the Data EEPROM Memory. To protect against spurious EEPROM writes, various mechanisms have been implemented. On power-up, the WREN bit is cleared. In addition, writes to the EEPROM are blocked during the Power-up Timer period (TPWRT).

The unlock sequence and the WREN bit together help prevent an accidental write during brown-out, power glitch or software malfunction.

## 13.3.8 ERASING THE DATA EEPROM MEMORY

Data EEPROM Memory can be erased by writing 0xFF to all locations in the Data EEPROM Memory that needs to be erased.

### EXAMPLE 13-7: DATA EEPROM REFRESH ROUTINE

|      |        |                |                              |
|------|--------|----------------|------------------------------|
|      | CLRF   | NVMADRL        | ; Start at address 0         |
|      | BCF    | NVMCON1, CFGS  | ; Set for memory             |
|      | BCF    | NVMCON1, EEPGD | ; Set for Data EEPROM        |
|      | BCF    | INTCON0, GIE   | ; Disable interrupts         |
|      | BSF    | NVMCON1, WREN  | ; Enable writes              |
| Loop |        |                | ; Loop to refresh array      |
|      | BSF    | NVMCON1, RD    | ; Read current address       |
|      | MOVLW  | 55h            | ;                            |
|      | MOVWF  | NVMCON2        | ; Write 55h                  |
|      | MOVLW  | 0AAh           | ;                            |
|      | MOVWF  | NVMCON2        | ; Write 0AAh                 |
|      | BSF    | NVMCON1, WR    | ; Set WR bit to begin write  |
|      | BTFSZ  | NVMCON1, WR    | ; Wait for write to complete |
|      | BRA    | \$-2           |                              |
|      | INCFSZ | NVMADRL, F     | ; Increment address          |
|      | BRA    | LOOP           | ; Not zero, do it again      |
|      | BCF    | NVMCON1, WREN  | ; Disable writes             |
|      | BSF    | INTCON0, GIE   | ; Enable interrupts          |

## 13.4 Register Definitions: Nonvolatile Memory

**REGISTER 13-1: NVMCON1: NONVOLATILE MEMORY CONTROL 1 REGISTER**

| R/W-0/0  | R/W-0/0 | U-0  | R/S/HC-0/0 | R/W/HS-x/q | R/W-0/0 | R/S/HC-0/0 | R/S/HC-0/0 |
|----------|---------|------|------------|------------|---------|------------|------------|
| REG<1:0> | —       | FREE | WRERR      | WREN       | WR      | RD         |            |
| bit 7    |         |      |            |            |         |            | bit 0      |

### Legend:

|                      |                   |                                                 |
|----------------------|-------------------|-------------------------------------------------|
| R = Readable bit     | W = Writable bit  | HC = Bit is cleared by hardware                 |
| x = Bit is unknown   | -n = Value at POR | S = Bit can be set by software, but not cleared |
| '0' = Bit is cleared | '1' = Bit is set  | U = Unimplemented bit, read as '0'              |

- bit 7-6 **REG<1:0>**: NVM Region Selection bit  
 10 = Access PFM Locations  
 x1 = Access User IDs, Configuration Bits, DIA, DCI, Rev ID and Device ID  
 00 = Access Data EEPROM Memory Locations
- bit 5 **Unimplemented**: Read as '0'
- bit 4 **FREE**: Program Flash Memory Erase Enable bit<sup>(1)</sup>  
 1 = Performs an erase operation on the next WR command  
 0 = The next WR command performs a write operation
- bit 3 **WRERR**: Write-Reset Error Flag bit<sup>(2,3,4)</sup>  
 1 = A write operation was interrupted by a Reset (hardware set),  
 or WR was written to 1'b1 when an invalid address is accessed (Table 4-1, Table 13-1)  
 or WR was written to 1'b1 when REG<1:0> and address do not point to the same region  
 or WR was written to 1'b1 when a write-protected address is accessed (Table 4-2).  
 0 = All write operations have completed normally
- bit 2 **WREN**: Program/Erase Enable bit  
 1 = Allows program/erase and refresh cycles  
 0 = Inhibits programming/erasing and user refresh of NVM
- bit 1 **WR**: Write Control bit<sup>(5,6,7)</sup>  
When REG points to a Data EEPROM Memory location:  
 1 = Initiates an erase/program cycle at the corresponding Data EEPROM Memory location  
When REG points to a PFM location:  
 1 = Initiates the PFM write operation with data from the holding registers  
 0 = NVM program/erase operation is complete and inactive
- bit 0 **RD**: Read Control bit<sup>(8)</sup>  
 1 = Initiates a read at address pointed by REG and NVMADR, and loads data into NVMDAT  
 0 = NVM read operation is complete and inactive

- Note 1:** This can only be used with PFM.
- 2:** This bit is set when WR = 1 and clears when the internal programming timer expires or the write is completed successfully.
- 3:** Bit must be cleared by the user; hardware will not clear this bit.
- 4:** Bit may be written to '1' by the user in order to implement test sequences.
- 5:** This bit can only be set by following the unlock sequence of [Section 13.1.4 "NVM Unlock Sequence"](#).
- 6:** Operations are self-timed and the WR bit is cleared by hardware when complete.
- 7:** Once a write operation is initiated, setting this bit to zero will have no effect.
- 8:** The bit can only be set in software. The bit is cleared by hardware when the operation is complete.



## REGISTER 13-2: NVMCON2: NONVOLATILE MEMORY CONTROL 2 REGISTER

|              |       |       |       |       |       |       |       |
|--------------|-------|-------|-------|-------|-------|-------|-------|
| R/W-0        | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| NVMCON2<7:0> |       |       |       |       |       |       |       |
| bit 7        |       |       |       |       |       |       | bit 0 |

### Legend:

R = Readable bit                      W = Writable bit                      U = Unimplemented bit, read as '0'  
 x = Bit is unknown                      '0' = Bit is cleared                      '1' = Bit is set  
 -n = Value at POR

bit 7-0                      **NVMCON2<7:0>:**  
 Refer to [Section 13.1.4 “NVM Unlock Sequence”](#).

**Note 1:** This register always reads zeros, regardless of data written.

## Register 13-3: NVMADRL: Data EEPROM Memory Address Low

|          |         |         |         |         |         |         |         |
|----------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0  | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| ADR<7:0> |         |         |         |         |         |         |         |
| bit 7    |         |         |         |         |         |         | bit 0   |

### Legend:

R = Readable bit                      W = Writable bit                      U = Unimplemented bit, read as '0'  
 x = Bit is unknown                      '0' = Bit is cleared                      '1' = Bit is set  
 -n = Value at POR

bit 7-0                      **ADR<7:0>:** EEPROM Read Address bits

## REGISTER 13-4: NVMADRH: DATA EEPROM MEMORY ADDRESS HIGH

|       |     |     |     |     |     |          |         |
|-------|-----|-----|-----|-----|-----|----------|---------|
| U-0   | U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0/0  | R/W-0/0 |
| —     | —   | —   | —   | —   | —   | ADR<9:8> |         |
| bit 7 |     |     |     |     |     |          | bit 0   |

### Legend:

R = Readable bit                      W = Writable bit                      U = Unimplemented bit, read as '0'  
 x = Bit is unknown                      '0' = Bit is cleared                      '1' = Bit is set  
 -n = Value at POR

bit 7-2                      **Unimplemented:** Read as '0'  
 bit 1-0                      **ADR<9:8>:** EEPROM Read Address bits

**REGISTER 13-5: NVMDAT: DATA EEPROM MEMORY DATA**

|          |         |         |         |         |         |         |         |
|----------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0  | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| DAT<7:0> |         |         |         |         |         |         |         |
| bit 7    |         |         |         |         |         |         |         |
| bit 0    |         |         |         |         |         |         |         |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

x = Bit is unknown

'0' = Bit is cleared

'1' = Bit is set

-n = Value at POR

bit 7-0

**DAT<7:0>:** The value of the data memory word returned from NVMADR after a Read command, or the data written by a Write command.

**TABLE 13-5: SUMMARY OF REGISTERS ASSOCIATED WITH NONVOLATILE MEMORY CONTROL**

| Name    | Bit 7          | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1       | Bit 0 | Register on Page    |
|---------|----------------|-------|-------|-------|-------|-------|-------------|-------|---------------------|
| NVMCON1 | REG<1:0>       |       | —     | FREE  | WRERR | WREN  | WR          | RD    | <a href="#">200</a> |
| NVMCON2 | Unlock Pattern |       |       |       |       |       |             |       | <a href="#">201</a> |
| NVMADRL | NVMADR<7:0>    |       |       |       |       |       |             |       | <a href="#">201</a> |
| NVMADRH | —              | —     | —     | —     | —     | —     | NVMADR<9:8> |       | <a href="#">201</a> |
| NVMDAT  | NVMDAT<7:0>    |       |       |       |       |       |             |       | <a href="#">202</a> |

**Legend:** — = unimplemented, read as '0'. Shaded bits are not used during EEPROM access.

\*Page provides register information.

## 14.0 CYCLIC REDUNDANCY CHECK (CRC) MODULE WITH MEMORY SCANNER

The Cyclic Redundancy Check (CRC) module provides a software-configurable hardware-implemented CRC checksum generator. This module includes the following features:

- Any standard CRC up to 16 bits can be used
- Configurable Polynomial
- Any seed value up to 16 bits can be used
- Standard and reversed bit order available
- Augmented zeros can be added automatically or by the user
- Memory scanner for fast CRC calculations on program/Data EEPROM memory user data
- Software loadable data registers for communication CRC's

### 14.1 CRC Module Overview

The CRC module provides a means for calculating a check value of program/Data EEPROM memory. The CRC module is coupled with a memory scanner for faster CRC calculations. The memory scanner can automatically provide data to the CRC module. The CRC module can also be operated by directly writing data to SFRs, without using a scanner.

## 14.2 CRC Functional Overview

The CRC module can be used to detect bit errors in the program memory using the built-in memory scanner or through user input RAM memory. The CRC module can accept up to a 16-bit polynomial with up to a 16-bit seed value. A CRC calculated check value (or checksum) will then be generated into the CRCACC<15:0> registers for user storage. The CRC module uses an XOR shift register implementation to perform the polynomial division required for the CRC calculation.

### EXAMPLE 14-1: CRC EXAMPLE

Rev. 10-000206A  
1/8/2014

**CRC-16-ANSI**

$x^{16} + x^{15} + x^2 + 1$  (17 bits)

Standard 16-bit representation = 0x8005

CRCXORH = 0b10000000  
CRCXORL = 0b0000010- <sup>(1)</sup>

**Data Sequence:**  
0x55, 0x66, 0x77, 0x88

DLEN = 0b0111  
PLEN = 0b1111

**Data entered into the CRC:**  
**SHIFTM = 0:**  
01010101 01100110 01110111 10001000

**SHIFTM = 1:**  
10101010 01100110 11101110 00010001

**Check Value (ACCM = 1):**  
**SHIFTM = 0:** 0x32D6  
CRCACCH = 0b00110010  
CRCACCL = 0b11010110

**SHIFTM = 1:** 0x6BA2  
CRCACCH = 0b01101011  
CRCACCL = 0b10100010

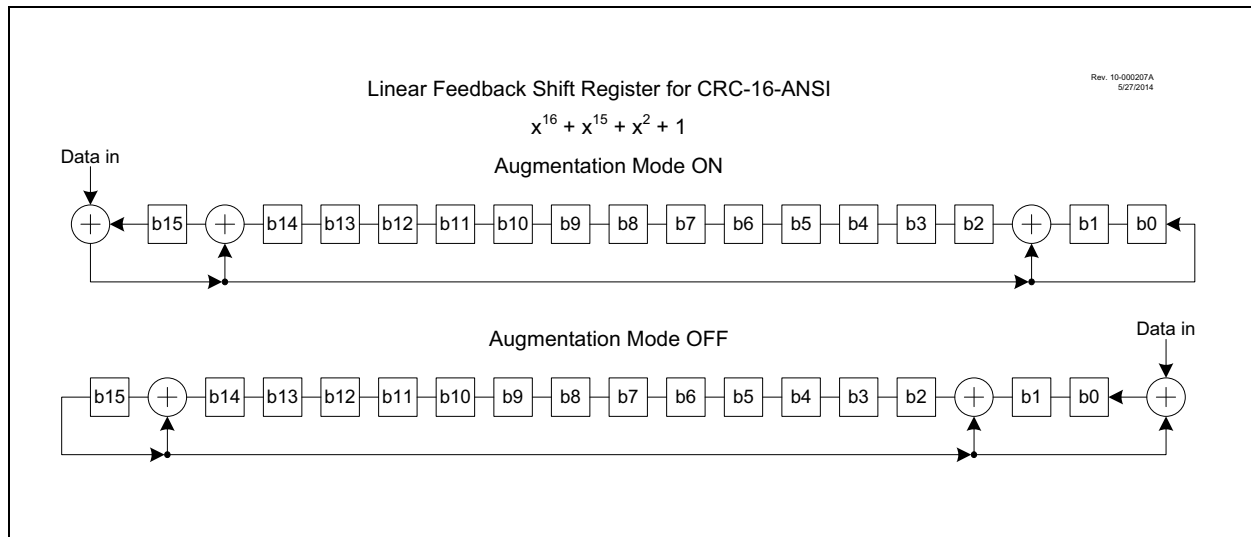
**Note 1:** Bit 0 is unimplemented. The LSb of any CRC polynomial is always '1' and will always be treated as a '1' by the CRC for calculating the CRC check value. This bit will be read in software as a '0'.

## 14.3 CRC Polynomial Implementation

Any polynomial can be used. The polynomial and accumulator sizes are determined by the  $PLEN<3:0>$  bits. For an  $n$ -bit accumulator,  $PLEN = n-1$  and the corresponding polynomial is  $n+1$  bits. Therefore the accumulator can be any size up to 16 bits with a corresponding polynomial up to 17 bits. The MSb and LSb of the polynomial are always '1' which is forced by hardware. All polynomial bits between the MSb and LSb are specified by the CRCXOR registers. For example, when using CRC-16-ANSI, the polynomial is defined as  $X^{16} + X^{15} + X^2 + 1$ .

The  $X^{16}$  and  $X^0 = 1$  terms are the MSb and LSb controlled by hardware. The  $X^{15}$  and  $X^2$  terms are specified by setting the corresponding  $CRCXOR<15:0>$  bits with the value of '0x8004'. The actual value is '0x8005' because the hardware sets the LSb to 1. However, the LSb of the CRCXORL register is unimplemented and always reads as '0'. Refer to [Example 14-1](#).

### EXAMPLE 14-2: CRC LFSR EXAMPLE



## 14.4 CRC Data Sources

Data can be input to the CRC module in two ways:

- User data using the CRCDAT registers (CRCDATH and CRCDATL)
- Program memory using the Program Memory Scanner

To set the number of bits of data, up to 16 bits, the  $DLEN$  bits of  $CRCCON1$  must be set accordingly. Only data bits in  $CRCDAT$  registers up to  $DLEN$  will be used, other data bits in  $CRCDAT$  registers will be ignored.

Data is moved into the  $CRCSHIFT$  as an intermediate to calculate the check value located in the  $CRCACC$  registers.

The  $SHIFTM$  bit is used to determine the bit order of the data being shifted into the accumulator. If  $SHIFTM$  is not set, the data will be shifted in MSb first (Big Endian). The value of  $DLEN$  will determine the MSb. If  $SHIFTM$  bit is set, the data will be shifted into the accumulator in reversed order, LSb first (Little Endian).

The CRC module can be seeded with an initial value by setting the  $CRCACC<15:0>$  registers to the appropriate value before beginning the CRC.

### 14.4.1 CRC FROM USER DATA

To use the CRC module on data input from the user, the user must write the data to the  $CRCDAT$  registers. The data from the  $CRCDAT$  registers will be latched into the shift registers on any write to the  $CRCDATL$  register.

### 14.4.2 CRC FROM FLASH

To use the CRC module on data located in Program memory, the user can initialize the Program Memory Scanner as defined in [Section 14.8, Scanner Module Overview](#).

## 14.5 CRC Check Value

The CRC check value will be located in the CRCACC registers after the CRC calculation has finished. The check value will depend on two mode settings of the CRCCON0 register: ACCM and SHIFTM. When the ACCM bit is set, the CRC module augments the data with a number of zeros equal to the length of the polynomial to align the final check value. When the ACCM bit is not set, the CRC will stop at the end of the data. A number of zeros equal to the length of the polynomial can then be entered into CRCDAT to find the same check value as augmented mode. Alternatively the expected check value can be entered at this point to make the final result equal '0'.

When the CRC check value is computed with the SHIFTM bit set, selecting LSb first, and the ACCM bit is also set then the final value in the CRCACC registers will be reversed such that the LSb will be in the MSb position and vice versa. This is the expected check value in bit reversed form. If you are creating a check value to be appended to a data stream then a bit reversal must be performed on the final value to achieve the correct checksum. You can use the CRC to do this reversal by the following method:

- Save the CRCACC value in user RAM space
- Clear the CRCACC registers
- Clear the CRCXOR registers
- Write the saved CRCACC value to the CRCDAT input.

The properly oriented check value will be in the CRCACC registers as the result.

## 14.6 CRC Interrupt

The CRC will generate an interrupt when the BUSY bit transitions from 1 to 0. The CRCIF Interrupt Flag is set every time the BUSY bit transitions, regardless of whether or not the CRC interrupt is enabled. The CRCIF bit can only be cleared in software.

## 14.7 Configuring the CRC

The following steps illustrate how to properly configure the CRC.

1. Determine if the automatic program memory scan will be used with the scanner or manual calculation through the SFR interface and perform the actions specified in [Section 14.4 "CRC Data Sources"](#), depending on which decision was made.
2. If desired, seed a starting CRC value into the CRCACCH/L registers.
3. Program the CRCXORH/L registers with the desired generator polynomial.
4. Program the DLEN<3:0> bits of the CRCCON1 register with the length of the data word - 1 (refer to [Example 14-1](#)). This determines how many times the shifter will shift into the accumulator for each data word.
5. Program the PLEN<3:0> bits of the CRCCON1 register with the length of the polynomial - 2 (refer to [Example 14-1](#)).
6. Determine whether shifting in trailing zeros is desired and set the ACCM bit of the CRCCON0 register appropriately.
7. Likewise, determine whether the MSb or LSb should be shifted first and write the SHIFTM bit of the CRCCON0 register appropriately.
8. Write the GO bit of the CRCCON0 register to begin the shifting process.
- 9a. If manual SFR entry is used, monitor the FULL bit of the CRCCON0 register. When FULL = 0, another word of data can be written to the CRCDATH/L registers, keeping in mind that CRCDATH should be written first if the data has more than eight bits, as the shifter will begin upon the CRCDATL register being written.
- 9b. If the scanner is used, the scanner will automatically load words into the CRCDATH/L registers as needed, as long as the GO bit is set.
- 10a. If manual entry is used, monitor the CRCIF (and BUSY bit to determine when the completed CRC calculation can be read from CRCACCH/L registers.
- 10b. If using the memory scanner, monitor the SCANIF (or the GO bit) for the scanner to finish pushing information into the CRCDAT registers. After the scanner is completed, monitor the BUSY bit to determine that the CRC has been completed and the check value can be read from the CRCACC registers. If both the interrupt flags are set and the BUSY and GO bits are cleared, the completed CRC calculation can be read from the CRCACCH/L registers.

## 14.8 Scanner Module Overview

The Scanner allows segments of the Program Flash Memory or Data EEPROM, to be read out (scanned) to the CRC Peripheral. The Scanner module interacts with the CRC module and supplies it data one word at a time. Data is fetched from the address range defined by SCANLADR registers up to the SCANHADR registers.

The Scanner begins operation when the SGO bit is set (SCANCON0 Register) and ends when either SGO is cleared by the user or when SCANLADR increments past SCANHADR. The SGO bit is also cleared by clearing the EN bit (CRCCON0 register).

## 14.9 Configuring the Scanner

The scanner module may be used in conjunction with the CRC module to perform a CRC calculation over a range of program memory or Data EEPROM addresses. In order to set up the scanner to work with the CRC, perform the following steps:

1. Set up the CRC module (See [Section 14.7 “Configuring the CRC”](#)) and enable the Scanner module by setting the EN bit in the SCANCON0 register.
2. Choose which memory region the Scanner module should operate on and set the MREG bit of the SCANCON0 register appropriately.
3. If trigger is used for scanner operation, set the TRIGEN bit of the SCANCON0 register and select the trigger source using SCANTRIG register. Select the trigger source using SCANTRIG register and then set the TRIGEN bit of the SCANCON0 register. See [Table 14-1](#) for Scanner Operation.
4. If Burst mode of operation is desired, set the BURSTMD bit (SCANCON0 register). See [Table 14-1](#) for Scanner Operation.
5. Set the SCANLADRL/H/U and SCANHADRL/H/U registers with the beginning and ending locations in memory that are to be scanned.
6. Select the priority level for the Scanner module (See [Section 3.1 “System Arbitration”](#)) and lock the priorities (See [Section 3.1.1 “Priority Lock”](#)).
7. Both CRCEN and CRCGO bits must be enabled to use the scanner. Setting the SGO bit will start the scanner operation.

## 14.10 Scanner Interrupt

The scanner will trigger an interrupt when the SCANLADR increments past SCANHADR. The SCANIF bit can only be cleared in software.

## 14.11 Scanning Modes

The interaction of the scanner with the system operation is controlled by the priority selection in the System Arbiter (see [Section 3.2 “Memory Access Scheme”](#)). Additionally, BURSTMD and TRIGEN also determine the operation of the Scanner.

### 14.11.1 TRIGEN = 0, BURSTMD = 0

In this case, the memory access request is granted to the scanner if no other higher priority source is requesting access.

All sources with lower priority than the scanner will get the memory access cycles that are not utilized by the scanner.

### 14.11.2 TRIGEN = 1, BURSTMD = 0

In this case, the memory access request is generated when the CRC module is ready to accept.

The memory access request is granted to the scanner if no other higher priority source is requesting access. All sources with lower priority than the scanner will get the memory access cycles that are not utilized by the scanner.

The memory access request is granted to the scanner if no other higher priority source is requesting access. All sources with lower priority than the scanner will get the memory access cycles that are not utilized by the scanner.

### 14.11.3 TRIGEN = x, BURSTMD = 1

In this case, the memory access is always requested by the scanner.

The memory access request is granted to the scanner if no other higher priority source is requesting access. The memory access cycles will not be granted to lower priority sources than the scanner until it completes operation i.e., SGO = 0 (SCANCON0 register)

**Note:** If TRIGEN = 1 and BURSTMD = 1, the user should ensure that the trigger source is active for the Scanner operation to complete.

## 14.12 Register Definitions: CRC and Scanner Control

Long bit name prefixes for the CRC and Scanner peripherals are shown below. Refer to [Section 1.3.2.2 “Long Bit Names”](#) for more information.

| Peripheral | Bit Name Prefix |
|------------|-----------------|
| CRC        | CRC             |

### REGISTER 14-1: CRCCON0: CRC CONTROL REGISTER 0

| R/W-0/0 | R/W-0/0 | R-0  | R/W-0/0 | U-0 | U-0 | R/W-0/0 | R-0   |
|---------|---------|------|---------|-----|-----|---------|-------|
| EN      | GO      | BUSY | ACCM    | —   | —   | SHIFTM  | FULL  |
| bit 7   |         |      |         |     |     |         | bit 0 |

#### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as ‘0’                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set     | ‘0’ = Bit is cleared |                                                       |

|         |                                                                                                                                                   |
|---------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| bit 7   | <b>EN:</b> CRC Enable bit<br>1 = CRC module is enabled<br>0 = CRC is disabled                                                                     |
| bit 6   | <b>GO:</b> CRC Go bit<br>1 = Start CRC serial shifter<br>0 = CRC serial shifter turned off                                                        |
| bit 5   | <b>BUSY:</b> CRC Busy bit<br>1 = Shifting in progress or pending<br>0 = All valid bits in shifter have been shifted into accumulator              |
| bit 4   | <b>ACCM:</b> Accumulator Mode bit<br>1 = Data is concatenated with zeros<br>0 = Data is not concatenated with zeros                               |
| bit 3-2 | <b>Unimplemented:</b> Read as ‘0’                                                                                                                 |
| bit 1   | <b>SHIFTM:</b> Shift Mode bit<br>1 = Shift right (LSb)<br>0 = Shift left (MSb)                                                                    |
| bit 0   | <b>FULL:</b> Data Path Full Indicator bit<br>1 = CRCDATH/L registers are full<br>0 = CRCDATH/L registers have shifted their data into the shifter |

### REGISTER 14-2: CRCCON1: CRC CONTROL REGISTER 1

| R/W-0/0   | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0   | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|-----------|---------|---------|---------|-----------|---------|---------|---------|
| DLEN<3:0> |         |         |         | PLEN<3:0> |         |         |         |
| bit 7     |         |         |         |           |         |         | bit 0   |

#### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as ‘0’                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set     | ‘0’ = Bit is cleared |                                                       |

|         |                                                                                                                               |
|---------|-------------------------------------------------------------------------------------------------------------------------------|
| bit 7-4 | <b>DLEN&lt;3:0&gt;:</b> Data Length bits<br>Denotes the length of the data word -1 (See <a href="#">Example 14-1</a> )        |
| bit 3-0 | <b>PLEN&lt;3:0&gt;:</b> Polynomial Length bits<br>Denotes the length of the polynomial -1 (See <a href="#">Example 14-1</a> ) |



## REGISTER 14-3: CRCDATH: CRC DATA HIGH BYTE REGISTER

|            |         |         |         |         |         |         |         |
|------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-xx     | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x |
| DATA<15:8> |         |         |         |         |         |         |         |
| bit 7      |         |         |         | bit 0   |         |         |         |

### Legend:

R = Readable bit                      W = Writable bit                      U = Unimplemented bit, read as '0'  
 u = Bit is unchanged                  x = Bit is unknown                      -n/n = Value at POR and BOR/Value at all other Resets  
 '1' = Bit is set                          '0' = Bit is cleared

bit 7-0                      **DATA<15:8>**: CRC Input/Output Data bits

## REGISTER 14-4: CRCDATL: CRC DATA LOW BYTE REGISTER

|           |         |         |         |         |         |         |         |
|-----------|---------|---------|---------|---------|---------|---------|---------|
| R/W-xx    | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x |
| DATA<7:0> |         |         |         |         |         |         |         |
| bit 7     |         |         |         | bit 0   |         |         |         |

### Legend:

R = Readable bit                      W = Writable bit                      U = Unimplemented bit, read as '0'  
 u = Bit is unchanged                  x = Bit is unknown                      -n/n = Value at POR and BOR/Value at all other Resets  
 '1' = Bit is set                          '0' = Bit is cleared

bit 7-0                      **DATA<7:0>**: CRC Input/Output Data bits  
 Writing to this register fills the shifter.

## REGISTER 14-5: CRCACCH: CRC ACCUMULATOR HIGH BYTE REGISTER

|           |         |         |         |         |         |         |         |
|-----------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0   | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| ACC<15:8> |         |         |         |         |         |         |         |
| bit 7     |         |         |         | bit 0   |         |         |         |

### Legend:

R = Readable bit                      W = Writable bit                      U = Unimplemented bit, read as '0'  
 u = Bit is unchanged                  x = Bit is unknown                      -n/n = Value at POR and BOR/Value at all other Resets  
 '1' = Bit is set                          '0' = Bit is cleared

bit 7-0                      **ACC<15:8>**: CRC Accumulator Register bits

## REGISTER 14-6: CRCACCL: CRC ACCUMULATOR LOW BYTE REGISTER

|          |         |         |         |         |         |         |         |
|----------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0  | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| ACC<7:0> |         |         |         |         |         |         |         |
| bit 7    |         |         |         | bit 0   |         |         |         |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0 **ACC<7:0>**: CRC Accumulator Register bits

## REGISTER 14-7: CRCSHIFTH: CRC SHIFT HIGH BYTE REGISTER

|             |     |     |     |       |     |     |     |
|-------------|-----|-----|-----|-------|-----|-----|-----|
| R-0         | R-0 | R-0 | R-0 | R-0   | R-0 | R-0 | R-0 |
| SHIFT<15:8> |     |     |     |       |     |     |     |
| bit 7       |     |     |     | bit 0 |     |     |     |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0 **SHIFT<15:8>**: CRC Shifter Register bits  
Reading from this register reads the CRC Shifter.

## REGISTER 14-8: CRCSHIFTL: CRC SHIFT LOW BYTE REGISTER

|            |     |     |     |       |     |     |     |
|------------|-----|-----|-----|-------|-----|-----|-----|
| R-0        | R-0 | R-0 | R-0 | R-0   | R-0 | R-0 | R-0 |
| SHIFT<7:0> |     |     |     |       |     |     |     |
| bit 7      |     |     |     | bit 0 |     |     |     |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0 **SHIFT<7:0>**: CRC Shifter Register bits  
Reading from this register reads the CRC Shifter.

## REGISTER 14-9: CRCXORH: CRC XOR HIGH BYTE REGISTER

| R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x |
|---------|---------|---------|---------|---------|---------|---------|---------|
| X<15:8> |         |         |         |         |         |         |         |
| bit 7   |         |         |         |         |         |         |         |
| bit 0   |         |         |         |         |         |         |         |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-0 **X<15:8>**: XOR of Polynomial Term  $X^n$  Enable bits

## REGISTER 14-10: CRCXORL: CRC XOR LOW BYTE REGISTER

| R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | U-1   |
|---------|---------|---------|---------|---------|---------|---------|-------|
| X<7:1>  |         |         |         |         |         |         | —     |
| bit 7   |         |         |         |         |         |         | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-1 **X<7:1>**: XOR of Polynomial Term  $X^n$  Enable bits

bit 0 **Unimplemented**: Read as '1'

## REGISTER 14-11: SCANCON0: SCANNER ACCESS CONTROL REGISTER 0

|         |         |            |     |     |         |         |       |
|---------|---------|------------|-----|-----|---------|---------|-------|
| R/W-0/0 | R/W-0/0 | R/W/HC-0/0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R-0/0 |
| EN      | TRIGEN  | SGO        | —   | —   | MREG    | BURSTMD | BUSY  |
| bit 7   |         |            |     |     |         |         | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

HC = Bit is cleared by hardware

bit 7 **EN:** Scanner Enable bit<sup>(1)</sup>

1 = Scanner is enabled

0 = Scanner is disabled

bit 6 **TRIGEN:** Scanner Trigger Enable bit<sup>(2)</sup>

1 = Scanner trigger is enabled

0 = Scanner trigger is disabled

Refer [Table 14-1](#).

bit 5 **SGO:** Scanner GO bit<sup>(3, 4)</sup>

1 = When the CRC is ready, the Memory region set by the MREG bit will be accessed and data is passed to the CRC peripheral.

0 = Scanner operations will not occur

bit 4-3 **Unimplemented:** Read as '0'

bit 2 **MREG:** Scanner Memory Region Select bit<sup>(2)</sup>

1 = Scanner address points to Data EEPROM

0 = Scanner address points to Program Flash Memory

bit 1 **BURSTMD:** Scanner Burst Mode bit

1 = Memory access request to the CPU Arbiter is always true

0 = Memory access request to the CPU Arbiter is dependent on the CRC request and Trigger

Refer [Table 14-1](#).

bit 0 **BUSY:** Scanner Busy Indicator bit

1 = Scanner cycle is in process

0 = Scanner cycle is complete (or never started)

**Note 1:** Setting EN = 1 (SCANCON0 register) does not affect any other register content.

**2:** Scanner trigger selection can be set using the SCANTRIG register.

**3:** This bit can be cleared in software. It is cleared in hardware when LADR>HADR (and a data cycle is not occurring) or when CRCGO = 0 (CRCCON0 register).

**4:** CRCEN and CRCGO bits (CRCCON0 register) must be set before setting the SGO bit.

**TABLE 14-1: SCANNER OPERATING MODES<sup>(1)</sup>**

| TRIGEN | BURSTMD | Scanner Operation                                                                                                                                                                   |
|--------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0      | 0       | Memory access is requested when the CRC module is ready to accept data; the request is granted if no other higher priority source request is pending.                               |
| 1      | 0       | Memory access is requested when the CRC module is ready to accept data and trigger selection is true; the request is granted if no other higher priority source request is pending. |
| x      | 1       | Memory access is always requested, the request is granted if no other higher priority source request is pending.                                                                    |

**Note 1:** See [Section 3.1 “System Arbitration”](#) for Priority selection and [Section 3.2 “Memory Access Scheme”](#) for Memory Access Scheme.

**REGISTER 14-12: SCANLADRU: SCAN LOW ADDRESS UPPER BYTE REGISTER**

|       |     |                              |         |         |         |         |         |
|-------|-----|------------------------------|---------|---------|---------|---------|---------|
| U-0   | U-0 | R/W-0/0                      | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| —     | —   | LADR<21:16> <sup>(1,2)</sup> |         |         |         |         |         |
| bit 7 |     |                              |         |         |         |         | bit 0   |

**Legend:**

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as ‘0’                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set     | ‘0’ = Bit is cleared |                                                       |

bit 7-6 **Unimplemented:** Read as ‘0’

bit 5-0 **LADR<21:16>:** Scan Start/Current Address bits<sup>(1,2)</sup>

Upper bits of the current address to be fetched from, value increments on each fetch of memory.

**Note 1:** Registers SCANLADRU/H/L form a 22-bit value, but are not guarded for atomic or asynchronous access; registers should only be read or written while SGO = 0 (SCANCON0 register).

**2:** While SGO = 1 (SCANCON0 register), writing to this register is ignored.

**REGISTER 14-13: SCANLADRH: SCAN LOW ADDRESS HIGH BYTE REGISTER**

|                              |         |         |         |         |         |         |         |
|------------------------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0                      | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| LADR<15:8> <sup>(1, 2)</sup> |         |         |         |         |         |         |         |
| bit 7                        |         |         |         |         |         |         | bit 0   |

**Legend:**

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as ‘0’                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set     | ‘0’ = Bit is cleared |                                                       |

bit 7-0 **LADR<15:8>:** Scan Start/Current Address bits<sup>(1, 2)</sup>

Most Significant bits of the current address to be fetched from, value increments on each fetch of memory.

**Note 1:** Registers SCANLADRU/H/L form a 22-bit value, but are not guarded for atomic or asynchronous access; registers should only be read or written while SGO = 0 (SCANCON0 register).

**2:** While SGO = 1 (SCANCON0 register), writing to this register is ignored.

## REGISTER 14-14: SCANLADRL: SCAN LOW ADDRESS LOW BYTE REGISTER

|                             |         |         |         |         |         |         |         |
|-----------------------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0                     | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| LADR<7:0> <sup>(1, 2)</sup> |         |         |         |         |         |         |         |
| bit 7                       |         |         |         |         |         |         | bit 0   |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0      **LADR<7:0>**: Scan Start/Current Address bits<sup>(1, 2)</sup>  
Least Significant bits of the current address to be fetched from, value increments on each fetch of memory

**Note 1:** Registers SCANLADRU/H/L form a 22-bit value, but are not guarded for atomic or asynchronous access; registers should only be read or written while SGO = 0 (SCANCON0 register).

**2:** While SGO = 1 (SCANCON0 register), writing to this register is ignored.

## REGISTER 14-15: SCANHADRU: SCAN HIGH ADDRESS UPPER BYTE REGISTER

|       |     |             |         |         |         |         |         |
|-------|-----|-------------|---------|---------|---------|---------|---------|
| U-0   | U-0 | R/W-1/1     | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
| —     | —   | HADR<21:16> |         |         |         |         |         |
| bit 7 |     |             |         |         |         |         | bit 0   |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-6      **Unimplemented:** Read as '0'

bit 5-0      **HADR<21:16>**: Scan End Address bits<sup>(1, 2)</sup>  
Upper bits of the address at the end of the designated scan

**Note 1:** Registers SCANHADRU/H/L form a 22-bit value but are not guarded for atomic or asynchronous access; registers should only be read or written while SGO = 0 (SCANCON0 register).

**2:** While SGO = 1 (SCANCON0 register), writing to this register is ignored.

## REGISTER 14-16: SCANHADR<sub>H</sub>: SCAN HIGH ADDRESS HIGH BYTE REGISTER

|                              |         |         |         |         |         |         |         |
|------------------------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-1/1                      | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
| HADR<15:8> <sup>(1, 2)</sup> |         |         |         |         |         |         |         |
| bit 7                        |         |         |         |         |         |         | bit 0   |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0      **HADR<15:8>**: Scan End Address bits<sup>(1, 2)</sup>  
Most Significant bits of the address at the end of the designated scan

- Note 1:** Registers SCANHADRU/H/L form a 22-bit value, but are not guarded for atomic or asynchronous access; registers should only be read or written while SGO = 0 (SCANCON0 register).  
**2:** While SGO = 1 (SCANCON0 register), writing to this register is ignored.

## REGISTER 14-17: SCANHADRL: SCAN HIGH ADDRESS LOW BYTE REGISTER

|                             |         |         |         |         |         |         |         |
|-----------------------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-1/1                     | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
| HADR<7:0> <sup>(1, 2)</sup> |         |         |         |         |         |         |         |
| bit 7                       |         |         |         |         |         |         | bit 0   |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0      **HADR<7:0>**: Scan End Address bits<sup>(1, 2)</sup>  
Least Significant bits of the address at the end of the designated scan

- Note 1:** Registers SCANHADRU/H/L form a 22-bit value, but are not guarded for atomic or asynchronous access; registers should only be read or written while SGO = 0 (SCANCON0 register).  
**2:** While SGO = 1 (SCANCON0 register), writing to this register is ignored.

## REGISTER 14-18: SCANTRIG: SCAN TRIGGER SELECTION REGISTER

| U-0   | U-0 | U-0 | U-0 | R/W-0/0   | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|-------|-----|-----|-----|-----------|---------|---------|---------|
| —     | —   | —   | —   | TSEL<3:0> |         |         |         |
| bit 7 |     |     |     | bit 0     |         |         |         |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-4

**Unimplemented:** Read as '0'

bit 3-0

**TSEL<3:0>:** Scanner Data Trigger Input Selection bits

1111 = Reserved

•  
•  
•

1010 = Reserved

1001 = SMT1\_output

1000 = TMR6\_postscaled

0111 = TMR5\_output

0110 = TMR4\_postscaled

0101 = TMR3\_output

0100 = TMR2\_postscaled

0011 = TMR1\_output

0010 = TMR0\_output

0001 = CLKREF\_output

0000 = LFINTOSC



**TABLE 14-2: SUMMARY OF REGISTERS ASSOCIATED WITH CRC**

| Name      | Bit 7       | Bit 6  | Bit 5       | Bit 4 | Bit 3     | Bit 2 | Bit 1   | Bit 0 | Register on Page |
|-----------|-------------|--------|-------------|-------|-----------|-------|---------|-------|------------------|
| CRCACCH   | ACC<15:8>   |        |             |       |           |       |         |       | 209              |
| CRCACCL   | ACC<7:0>    |        |             |       |           |       |         |       | 210              |
| CRCCON0   | EN          | GO     | BUSY        | ACCM  | —         | —     | SHIFTM  | FULL  | 208              |
| CRCCON1   | DLEN<3:0>   |        |             |       | PLEN<3:0> |       |         |       | 208              |
| CRCDATH   | DATA<15:8>  |        |             |       |           |       |         |       | 209              |
| CRCDATL   | DATA<7:0>   |        |             |       |           |       |         |       | 209              |
| CRCSHIFTH | SHIFT<15:8> |        |             |       |           |       |         |       | 210              |
| CRCSHIFTL | SHIFT<7:0>  |        |             |       |           |       |         |       | 210              |
| CRCXORH   | X<15:8>     |        |             |       |           |       |         |       | 211              |
| CRCXORL   | X<7:1>      |        |             |       |           |       |         | —     | 211              |
| SCANCON0  | EN          | TRIGEN | SGO         | —     | —         | MREG  | BURSTMD | BUSY  | 212              |
| SCANHADRU | —           | —      | HADR<21:16> |       |           |       |         |       | 214              |
| SCANHADRH | HADR<15:8>  |        |             |       |           |       |         |       | 215              |
| SCANHADRL | HADR<7:0>   |        |             |       |           |       |         |       | 215              |
| SCANLADRU | —           | —      | LADR<21:16> |       |           |       |         |       | 213              |
| SCANLADRH | LADR<15:8>  |        |             |       |           |       |         |       | 213              |
| SCANLADRL | LADR<7:0>   |        |             |       |           |       |         |       | 214              |
| SCANTRIG  | —           | —      | —           | —     | TSEL<3:0> |       |         |       | 216              |

**Legend:** — = unimplemented location, read as '0'. Shaded cells are not used for the CRC module.

## 15.0 DIRECT MEMORY ACCESS (DMA)

### 15.1 Introduction

The Direct Memory Access (DMA) module is designed to service data transfers between different memory regions directly without intervention from the CPU. By eliminating the need for CPU-intensive management of handling interrupts intended for data transfers, the CPU now can spend more time on other tasks.

PIC18(L)F25/26K83 family has two DMA modules which can be independently programmed to transfer data between different memory locations, move different data sizes, and use a wide range of hardware triggers to initiate transfers. The two DMA registers can even be programmed to work together, in order to carry out more complex data transfers without CPU overhead.

Key features of the DMA module include:

- Support access to the following memory regions:
  - GPR and SFR space (R/W)
  - Program Flash Memory (R only)
  - Data EEPROM Memory (R only)
- Programmable priority between the DMA and CPU Operations. Refer to [Section 3.1 “System Arbitration”](#) for details.
- Programmable Source and Destination address modes
  - Fixed address
  - Post-increment address
  - Post-decrement address
- Programmable Source and Destination sizes
- Source and destination pointer register, dynamically updated and reloadable
- Source and destination count register, dynamically updated and reloadable
- Programmable auto-stop based on Source or Destination counter
- Software triggered transfers
- Multiple user selectable sources for hardware triggered transfers
- Multiple user selectable sources for aborting DMA transfers

### 15.2 DMA Registers

The operation of the DMA module has the following registers:

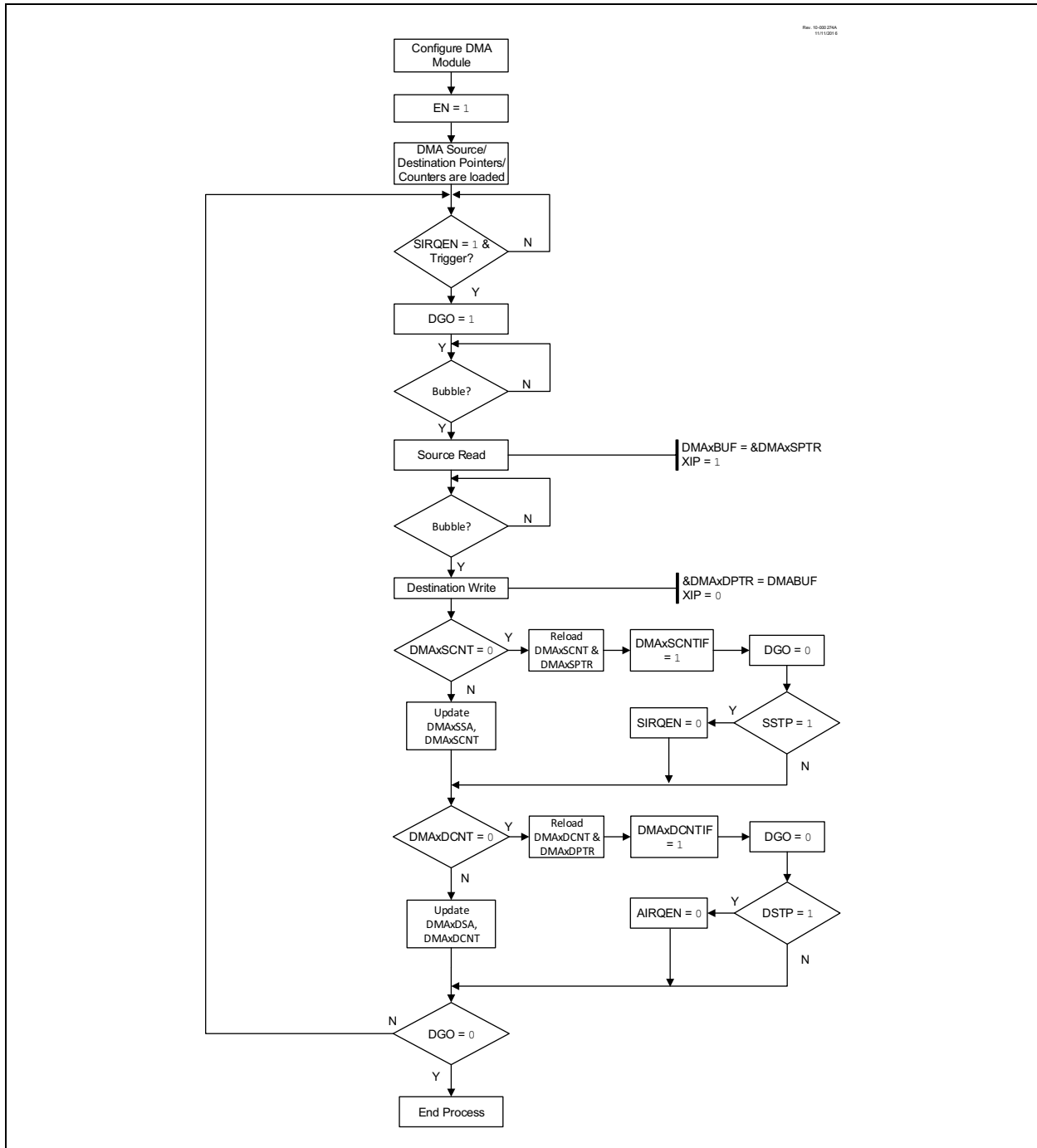
- Control registers (DMAxCON0, DMAxCON1)
- Data buffer register (DMAxBUF)
- Source Start Address Register (DMAxSSAU:H:L)
- Source Pointer Register (DMAxSPTRU:H:L)
- Source Message Size Register (DMAxSSZH:L)
- Source Count Register (DMAxSCNTH:L)
- Destination Start Address Register (DMAxDSAH:L)
- Destination Pointer Register (DMAxDPTRH:L)
- Destination Message Size Register (DMAxDSZH:L)
- Destination Count Register (DMAxDCNTH:L)
- Start Interrupt Request Source Register (DMAxSIRQ)
- Abort Interrupt Request Source Register (DMAxAIRQ)

These registers are detailed in [Section 15.13 “Register definitions: DMA”](#).

## 15.3 DMA Organization

The DMA module on the K83 family of devices is designed to move data by using the existing Instruction Bus<16> and Data Bus<8> without the need for any dual-porting of memory or peripheral systems (Figure 15-1). The DMA accesses the required bus when it has been granted to by the System Arbiter.

**FIGURE 15-1: DMA FUNCTIONAL BLOCK DIAGRAM**



Depending on the priority of the DMA with respect to CPU execution (Refer to [Section 3.2 “Memory Access Scheme”](#) for more information), the DMA Controller can move data through two methods:

- Stalling the CPU execution until it has completed its transfers (DMA has higher priority over the CPU in this mode of operation)
- Utilizing unused CPU cycles for DMA transfers (CPU has higher priority over the DMA in this mode of operation). Unused CPU cycles are referred to as bubbles which are instruction cycles available for use by the DMA to perform read and write operations. In this way, the effective bandwidth for handling data is increased; at the same time, DMA operations can proceed without causing a processor stall.

## 15.4 DMA Interface

The DMA module transfers data from the source to the destination one byte at a time, this smallest data movement is called a DMA data transaction. A DMA Message refers to one or more DMA data transactions.

Each DMA data transaction consists of two separate actions:

- Reading the Source Address Memory and storing the value in the DMA Buffer register
- Writing the contents of the DMA Buffer register to the Destination Address Memory

**Note:** DMA data movement is a two-cycle operation.

The XIP bit (DMAxCON0 register) is a Status bit to indicate whether or not the data in the DMAxBUF register has been written to the destination address. If the bit is set then data is waiting to be written to the destination. If clear it means that either data has been written to the destination or that no source read has occurred.

The DMA has read access to PFM, Data EEPROM, and SFR/GPR space, and write access to SFR/GPR space. Based on these memory access capabilities, the DMA can support the following memory transactions:

**TABLE 15-1: DMA MEMORY ACCESS**

| Read Source          | Write Destination |
|----------------------|-------------------|
| Program Flash Memory | GPR               |
| Program Flash Memory | SFR               |
| Data EE              | GPR               |
| Data EE              | SFR               |
| GPR                  | GPR               |
| SFR                  | GPR               |
| GPR                  | SFR               |
| SFR                  | SFR               |

Even though the DMA module has access to all memory and peripherals that are also available to the CPU, it is recommended that the DMA does not access any register that is part of the System arbitration. The DMA, as a system arbitration client should not be read or written by itself or by another DMA instantiation.

The following sections discuss the various control interfaces required for DMA data transfers.

### 15.4.1 DMA ADDRESSING

The start addresses for the source read and destination write operations are set using the DMAxSSA <21:0> and DMAxDSA <15:0> registers, respectively.

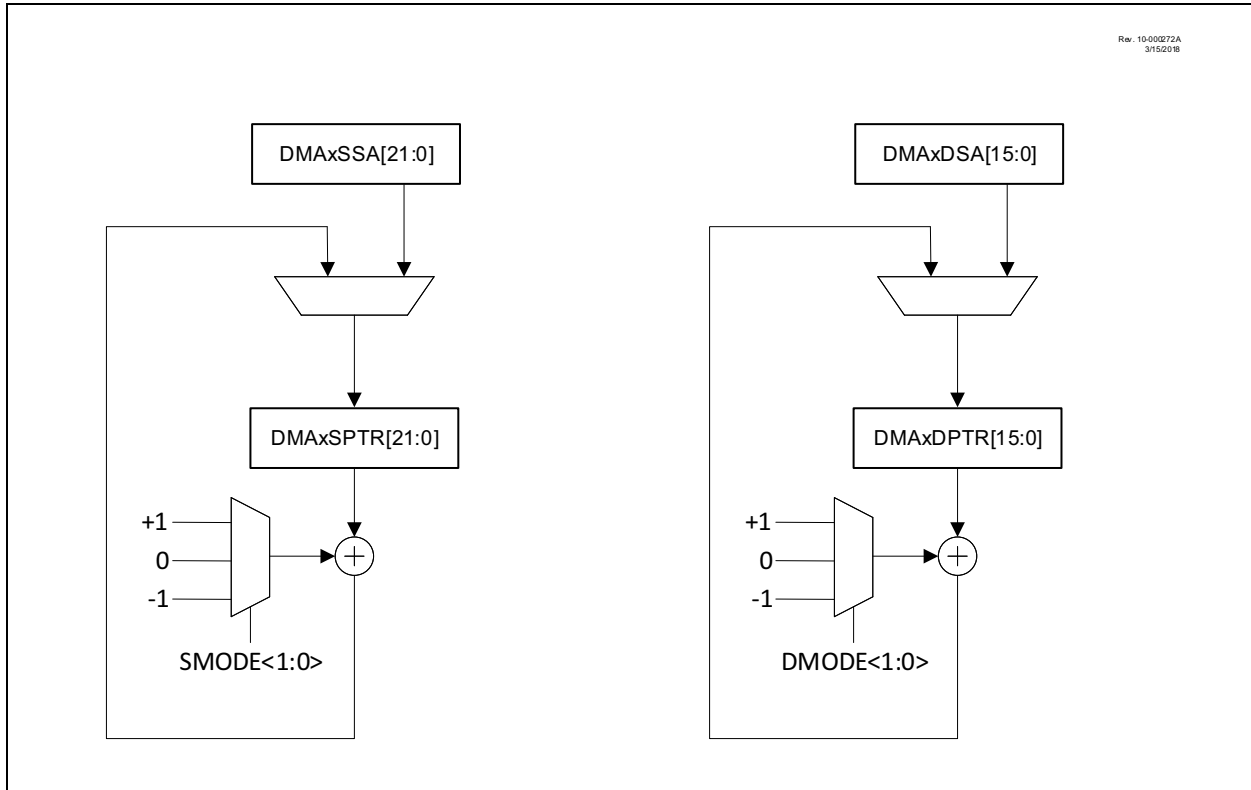
When the DMA Message transfers are in progress, the DMAxSPTR <21:0> and DMAxDPTR <15:0> registers contain the current address pointers for each source read and destination write operation, these registers are modified after each transaction based on the Address mode selection bits.

The SMODE and DMODE bits in the DMAxCON1 control register determine the address modes of operation by controlling how the DMAxSPTR <21:0> and DMAxDPTR <15:0> bits are updated after every DMA data transaction combination ([Figure 15-2](#)).

Each address can be separately configured to:

- Remain unchanged
- Increment by 1
- Decrement by 1

**FIGURE 15-2: DMA POINTERS BLOCK DIAGRAM**



The DMA can initiate data transfers from the PFM, Data EEPROM or SFR/GPR Space. The SMR<1:0> bits in the DMAxCON1 register are used to select the type of memory being pointed to by the Source Address Pointer. The SMR<1:0> bits are required because the PFM and SFR/GPR spaces have overlapping addresses that do not allow the specified address to uniquely define the memory location to be accessed.

registers count the destination transactions. Both are simultaneously decremented by one after each transaction.

- Note 1:** For proper memory read access to occur, the combination of address and space selection must be valid.
- 2:** The destination does not have space selection bits because it can only write to the SFR/GPR space.

## 15.4.2 DMA MESSAGE SIZE/COUNTERS

A transaction is the transfer of one byte. A message consists of one or more transactions. A complete DMA process consists of one or more messages. The size registers determine how many transactions are in a message. The DMAxSSZ registers determine the source size and DMAxDSZ registers determine the destination size.

When a DMA transfer is initiated, the size registers are copied to corresponding counter registers that control the duration of the message. The DMAxSCNT registers count the source transactions and the DMAxDCNT

A message is started by setting the DGO bit of the DMAxCON0 register and terminates when the smaller of the two counters reaches zero.

When either counter reaches zero the DGO bit is cleared and the counter and pointer registers are immediately reloaded with the corresponding size and address data. If the other counter did not reach zero then the next message will continue with the count and address corresponding to that register.

When the source and destination size registers are not equal then the ratio of the largest to the smallest size determines how many messages are in the DMA process. For example, when the destination size is 6 and the source size is 2 then each message will consist of two transactions and the complete DMA process will consist of three messages. When the larger size is not an even integer of the smaller size then the last message in the process will terminate early when the larger count reaches zero. In that case, the larger counter will reset and the smaller counter will have a remainder skewing any subsequent messages by that amount.

|                                                                                                                                                                                                      |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p><b>Note:</b> Reading the DMAxSCNT or DMAxDCNT registers will never return zero. When either register is decremented from '1' it is immediately reloaded from the corresponding size register.</p> |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**FIGURE 15-3: DMA COUNTERS BLOCK DIAGRAM**

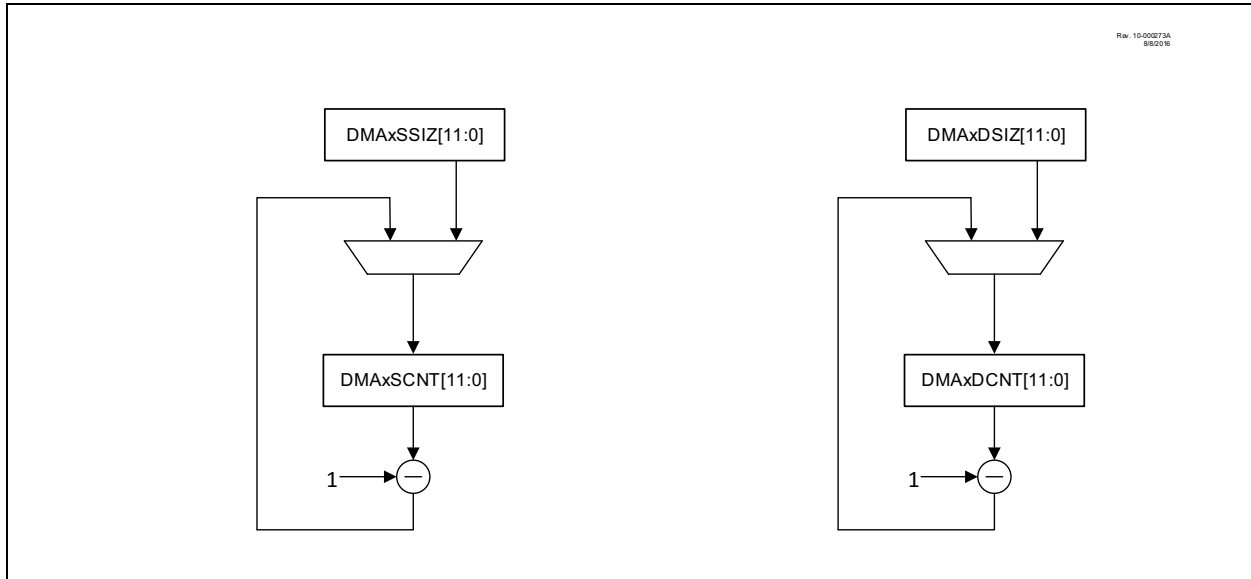


Table 15-2 has a few examples of configuring DMA Message sizes.

**TABLE 15-2: EXAMPLE MESSAGE SIZE TABLE**

| Operation                             | Example           | SCNT         | DCNT        | Comments                                                                                   |
|---------------------------------------|-------------------|--------------|-------------|--------------------------------------------------------------------------------------------|
| Read from single SFR location to RAM  | U1RXB             | 1            | N           | N equals the number of bytes desired in the destination buffer. $N \geq 1$ .               |
| Write to single SFR location from RAM | U1TXB             | N            | 1           | N equals the number of bytes desired in the source buffer. $N \geq 1$ .                    |
| Read from multiple SFR location       | ADRES[H:L]        | 2            | $2 \cdot N$ | N equals the number of ADC results to be stored in memory. $N \geq 1$                      |
|                                       | TMR1[H:L]         | 2            | $2 \cdot N$ | N equals the number of TMR1 Acquisition results to be stored in memory. $N \geq 1$         |
|                                       | SMT1CPR[U:H:L]    | 3            | $3 \cdot N$ | N equals the number of Capture Pulse Width measurements to be stored in memory. $N \geq 1$ |
| Write to Multiple SFR registers       | PWMDC[H:L]        | $2 \cdot N$  | 2           | N equals the number of PWM duty cycle values to be loaded from a memory table. $N \geq 1$  |
|                                       | All ADC registers | $N \cdot 31$ | 31          | Using the DMA to transfer a complete ADC context from RAM to the ADC registers. $N \geq 1$ |

## 15.5 DMA Message Transfers

Once the Enable bit is set to start DMA message transfers, the Source/Destination pointer and counter registers are initialized to the conditions shown in [Table 15-3](#).

**TABLE 15-3: DMA INITIAL CONDITIONS**

| Register       | Value loaded  |
|----------------|---------------|
| DMAxSPTR<21:0> | DMAxSSA<21:0> |
| DMAxSCNT<11:0> | DMAxSSZ<11:0> |
| DMAxDPTR<15:0> | DMAxDSA<15:0> |
| DMAxDCNT<11:0> | DMAxDSZ<11:0> |

During the DMA Operation after each transaction, [Table 15-4](#) and [Table 15-5](#) indicate how the Source/Destination pointer and counter registers are modified

**TABLE 15-4: DMA SOURCE POINTER/COUNTER DURING OPERATION**

| Register            | Modified Source Counter/Pointer Value |
|---------------------|---------------------------------------|
| DMAxSCNT<11:0> != 1 | DMAxSCNT = DMAxSCNT - 1               |
|                     | SMODE = 00: DMAxSPTR = DMAxSPTR       |
|                     | SMODE = 01: DMAxSPTR = DMAxSPTR + 1   |
|                     | SMODE = 10: DMAxSPTR = DMAxSPTR - 1   |
| DMAxSCNT<11:0> == 1 | DMAxSCNT = DMAxSSZ                    |
|                     | DMAxSPTR = DMAxSSA                    |

**TABLE 15-5: DMA DESTINATION POINTER/COUNTER DURING OPERATION**

| Register            | Modified Destination Counter/Pointer Value |
|---------------------|--------------------------------------------|
| DMAxDCNT<11:0> != 1 | DMAxDCNT = DMAxDCNT - 1                    |
|                     | DMODE = 00: DMAxDPTR = DMAxDPTR            |
|                     | DMODE = 01: DMAxDPTR = DMAxDPTR + 1        |
|                     | DMODE = 10: DMAxDPTR = DMAxDPTR - 1        |
| DMAxDCNT<11:0> == 1 | DMAxDCNT = DMAxDSZ                         |
|                     | DMAxDPTR = DMAxDSA                         |

The following sections discuss how to initiate and terminate DMA transfers.

### 15.5.1 STARTING DMA MESSAGE TRANSFERS

The DMA can initiate data transactions by either of the following two conditions:

1. User software control
2. Hardware trigger, SIRQ

#### 15.5.1.1 User Software Control

Software starts or stops DMA transaction by setting/clearing the DGO bit. The DGO bit is also used to indicate whether a DMA hardware trigger has been received and a message is in progress.

- Note 1:** Software start can only occur if the EN bit (DMAxCON1) is set.
- 2:** If the CPU writes to the DGO bit while it is already set, there is no effect on the system, the DMA will continue to operate normally.



## 15.5.1.2 Hardware Trigger, SIRQ

A Hardware trigger is an interrupt request from another module sent to the DMA with the purpose of starting a DMA message. The DMA start trigger source is user selectable using the DMAxSIRQ register.

The SIRQEN bit (DMAxCON0 register) is used to enable sampling of external interrupt triggers by which a DMA transfer can be started. When set the DMA will sample the selected Interrupt source and when cleared, the DMA will ignore the selected Interrupt source. Clearing SIRQEN does not stop a DMA transaction currently progress, it only stops more hardware request signals from being received.

## 15.5.2 STOPPING DMA MESSAGE TRANSFERS

The DMA controller can stop data transactions by either of the following two conditions:

1. Clearing the DGO bit
2. Hardware trigger, AIRQ
3. Source Count reload
4. Destination Count reload
5. Clearing the Enable bit

### 15.5.2.1 User Software Control

If the user clears the DGO bit, the message will be stopped and the DMA will remain in the current configuration.

For example, if the user clears the DGO bit after source data has been read but before it is written to the destination, then the data in DMAxBUF will not reach its destination.

This is also referred to as a soft-stop as the operation can resume if desired by setting DGO bit again.

### 15.5.2.2 Hardware Trigger, AIRQ

The AIRQEN bit (DMAxCON0 register) is used to enable sampling of external interrupt triggers by which a DMA transaction can be aborted.

Once an Abort interrupt request has been received, the DMA will perform a soft-stop by clearing the DGO bit as well as clearing the SIRQEN bit so overruns do not occur. The AIRQEN bit is also cleared to prevent additional abort signals from triggering false aborts.

If desired, the DGO bit can be set again and the DMA will resume operation from where it left off after the soft-stop had occurred as none of the DMA state information is changed in the event of an abort.

### 15.5.2.3 Source Count Reload

A DMA message is considered to be complete when the Source count register is decremented from 1 and then reloaded (i.e., once the last byte from either the source read or destination write has occurred). When the SSTP bit is set (DMAxCON1 register) and the source count register is reloaded then further message transfer is stopped.

### 15.5.2.4 Destination Count Reload

A DMA message is considered to be complete when the Destination count register is decremented from 1 and then reloaded (i.e., once the last byte from either the source read or destination write has occurred). When the DSTP bit is set (DMAxCON1) and the destination count register is reloaded then further message transfer is stopped.

**Note:** Reading the DMAxSCNT or DMAxDCNT registers will never return zero. When either register is decremented from '1' it is immediately reloaded from the corresponding size register.

### 15.5.2.5 Clearing the Enable bit

If the User clears the EN bit, the message will be stopped and the DMA will return to its default configuration. This is also referred to as a hard-stop as the DMA cannot resume operation from where it was stopped.

**Note:** After the DMA message transfer is stopped, it requires an extra instruction cycle before the Stop condition takes effect. Thus, after the Stop condition has occurred, a Source read or a Destination write can occur depending on the Source or Destination Bus availability.

## 15.5.3 DISABLE DMA MESSAGES TRANSFERS UPON COMPLETION

Once the DMA message is complete it may be desirable to disable the trigger source to prevent overrun or under run of data. This can be done by either of the following methods:

1. Clearing the SIRQEN bit
2. Setting the SSTP bit
3. Setting the DSTP bit

## 15.5.3.1 Clearing the SIRQEN bit

Clearing the SIRQEN bit (DMAxCON1 register) stops the sampling of external start interrupt triggers hence preventing further DMA Message transfers.

An example would be a communications peripheral with a level-triggered interrupt. The peripheral will continue to request data (because its buffer is empty) even though there is no more data to be moved. Disabling the SIRQEN bit prevents the DMA from processing these requests

## 15.5.3.2 Source/Destination Stop

The SSTP and DSTP bits (DMAxCON0 register) determine whether or not to disable the hardware triggers (SIRQEN = 0) once a DMA message has completed.

When the SSTP bit is set and the DMAxSCNT = 0, then the SIRQEN bit will be cleared. Similarly, when the DSTP bit is set and the DMAxDCNT = 0, the SIRQEN bit will be cleared.

**Note:** The SSTP and DSTP bits are independent functions and do not depend on each other. It is possible for a message to be stopped by either counter at message end or both counters at message end.

## 15.6 Types of Hardware Triggers

The DMA has two different trigger inputs namely the Source trigger and the abort trigger. Each of these trigger sources is user configurable using the DMAxSIRQ and DMAxAIRQ registers.

Based on the source selected for each trigger, there are two types of requests that can be sent to the DMA.

- Edge triggers
- Level triggers

### 15.6.1 EDGE TRIGGER REQUESTS

Edge triggers are generated by the signal that sets the corresponding interrupt flag. The DMA responds to this event but leaves the interrupt flag set. An Edge request occurs only once when a given module interrupt requirements are true.

### 15.6.2 LEVEL TRIGGER REQUESTS

A level request is asserted as long as the condition that causes the interrupt is true. For example, the RXIF interrupt is asserted as long as the UART receive buffer has unread data. The RXIF cannot be cleared except by emptying the receive buffer.

## 15.7 Types of Data Transfers

Based on the memory access capabilities of the DMA (See [Table 15-1](#)), the following sections discuss the different types of data movement between the Source and Destination Memory regions.

- N: 1

This type of transfer is common when sending predefined data packets (such as strings) through a single interface point (such as communications modules transmit registers).

- N: N

This type of transfer is useful for moving information out of the Program Flash or Data EEPROM to SRAM for manipulation by the CPU or other peripherals.

- 1: N

This type of transfer is common when bridging two different modules data streams together (communications bridge).

- 1: N

This type of transfer is useful for moving information from a single data source into a memory buffer (communications receive registers).

## 15.8 DMA Interrupts

Each DMA has its own set of four interrupt flags, used to indicate a range of conditions during data transfers. The interrupt flag bits can be accessed using the corresponding PIR registers (Refer to the Interrupt Section).

### 15.8.1 DMA SOURCE COUNT INTERRUPT

The DMAxSCNTIF source count interrupt flag is set every time the DMAxSCNT<11:0> reaches zero and is reloaded to its starting value.

### 15.8.2 DMA DESTINATION COUNT INTERRUPT

The DMAxDCNTIF destination count interrupt flag is set every time the DMAxDCNT<11:0> reaches zero and is reloaded to its starting value.

The DMA Source Count zero and Destination Count zero interrupts are used in conjunction to determine when to signal the CPU when the DMA Messages are completed.

### 15.8.3 ABORT INTERRUPT

The DMAxAIF abort interrupt flag is used to signal that the DMA has halted activity due to an abort signal from one of the abort sources. This is used to indicate that the transaction has been halted for some reason.

## 15.8.4 OVERRUN INTERRUPT

When the DMA receives a trigger to start a new message before the current message is completed, then the DMAxORIF Overrun interrupt flag is set.

This condition indicates that the DMA is being requested before its current transaction is finished. This implies that the active DMA may not be able to keep up with the demands from the peripheral module being serviced, which may result in data loss.

The DMAxORIF flag being set does not cause the current DMA transfer to terminate.

The Overrun interrupt is only available for trigger sources that are edge based and not available for sources that are level-based. Therefore a level-based interrupt source does not trigger a DMA overrun error due to the potential latency issues in the system.

An example of an interrupt that could use the overrun interrupt would be a timer overflow (or period match) interrupt. This event only happens every time the timer rolls over and is not dependent on any other system conditions.

An example of an interrupt that does not allow the overrun interrupt would be the UARTTX buffer. The UART will continue to assert the interrupt until the DMA is able to process the MSG. Due to latency issues, the DMA may not be able to service an empty buffer immediately, but the UART continues to assert its transmit interrupt until it is serviced. If overrun was allowed in this case, the overrun would occur almost immediately as the module samples the interrupt sources every instruction cycle.

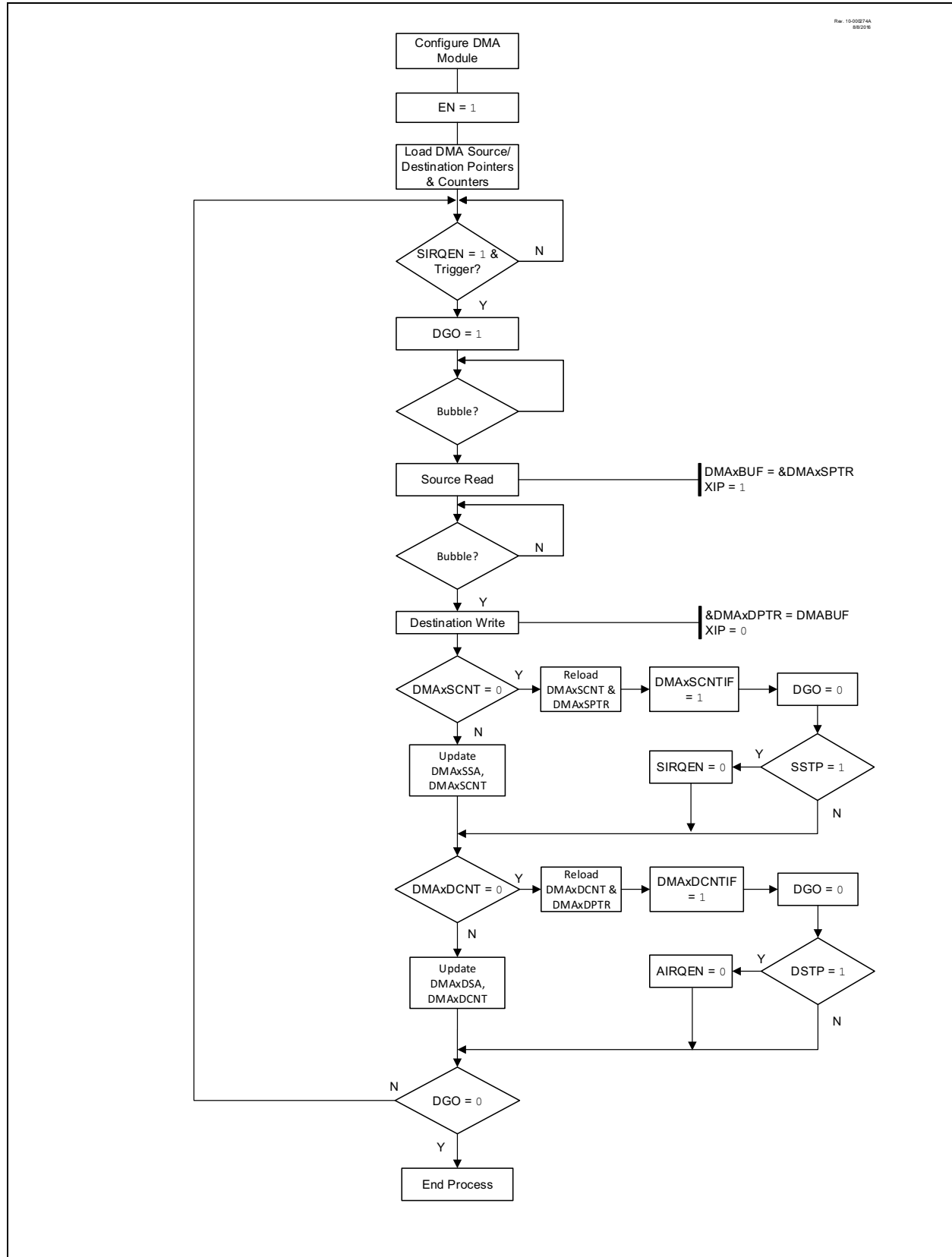
## 15.9 DMA Setup and Operation

The following steps illustrate how to configure the DMA for data transfer:

1. Program the appropriate Source and Destination addresses for the transaction into the DMAxSSA and DMAxDSA registers
2. Select the source memory region that is being addressed by DMAxSSA register, using the SMR<1:0> bits.
3. Program the SMODE and DMODE bits to select the addressing mode.
4. Program the Source size DMAxSSZ and Destination size DMAxDSZ registers with the number of bytes to be transferred. It is recommended for proper operation that the size registers be a multiple of each other.
5. If the user desires to disable data transfers once the message has completed, then the SSTP and DSTP bits in DMAxCON0 register need to be set. (see [Section 15.5.3.2 "Source/Destination Stop"](#)).
6. If using hardware triggers for data transfer, setup the hardware trigger interrupt sources for the starting and aborting DMA transfers (DMAxSIRQ and DMAxAIRQ), and set the corresponding interrupt request enable bits (SIRQEN and AIRQEN).
7. Select the priority level for the DMA (see [Section 3.1 "System Arbitration"](#)) and lock the priorities (see [Section 3.1.1 "Priority Lock"](#))
8. Enable the DMA (DMAxCON1bits. EN = 1)
9. If using software control for data transfer, set the DGO bit, else this bit will be set by the hardware trigger.

Once the DMA is set up, the following flow chart describes the sequence of operation when the DMA uses hardware triggers and utilizes the unused CPU cycles (bubble) for DMA transfers.

**FIGURE 15-4: DMA OPERATION WITH HARDWARE TRIGGER**

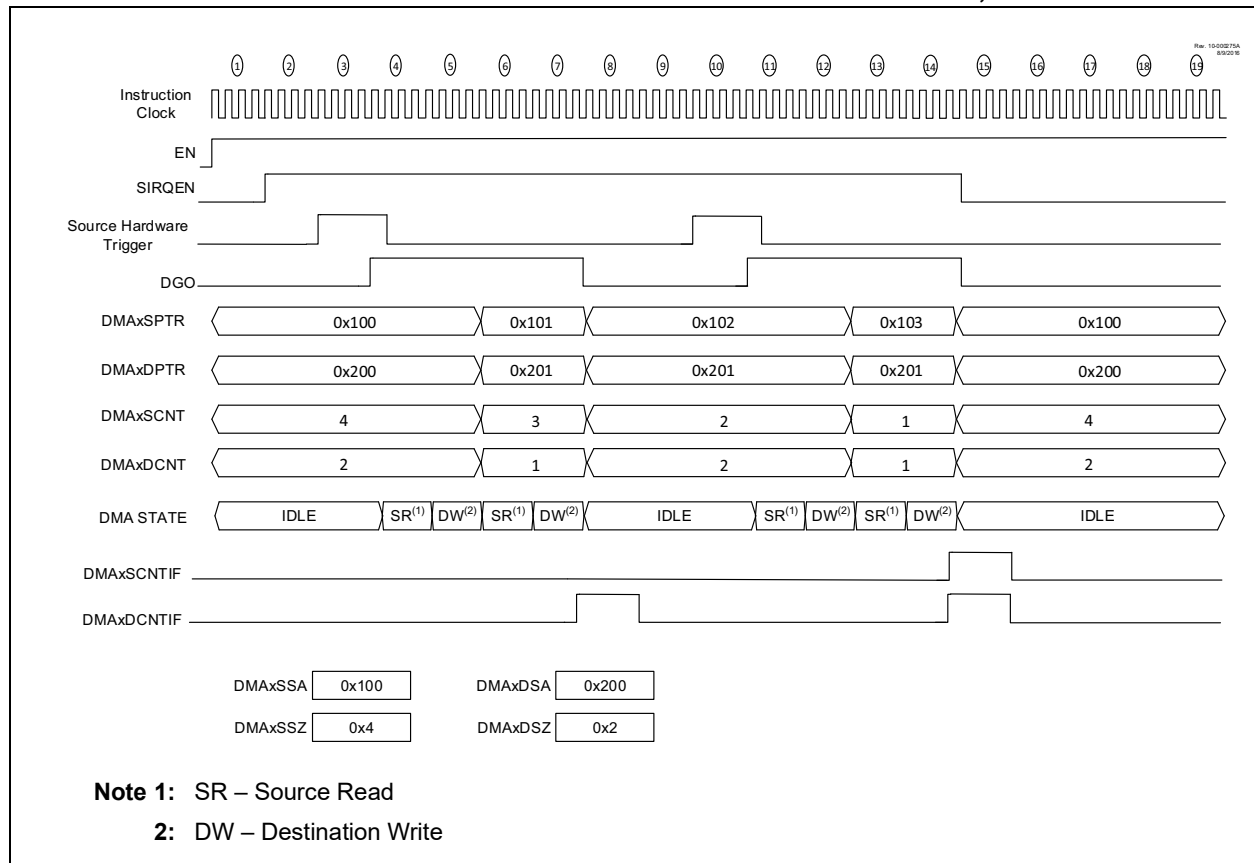


The following sections describe with visual reference the sequence of events for different configurations of the DMA module

## 15.9.1 SOURCE STOP

When the Source Stop bit is set (SSTP = 1) and the DMAxSCNT register reloads, the DMA clears the SIRQEN bit to stop receiving new start interrupt request signals and sets the DMAxSCNTIF flag.

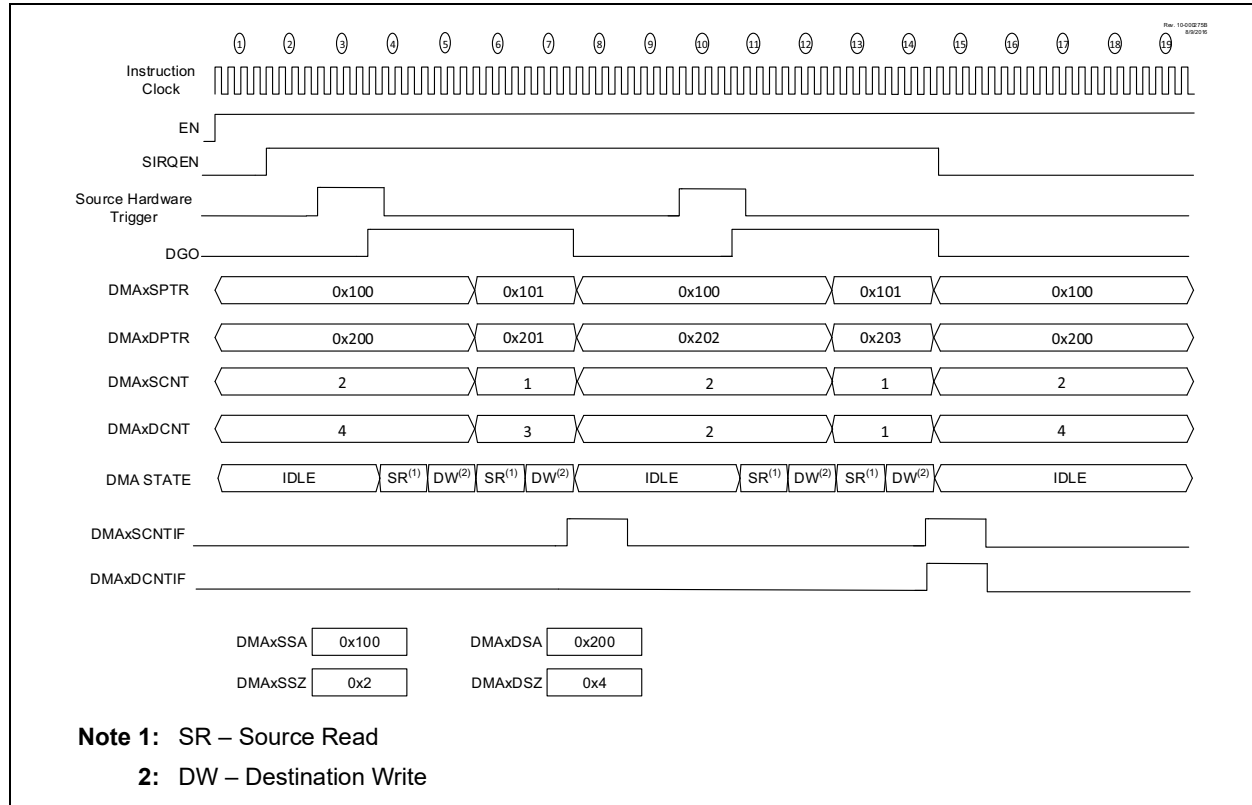
**FIGURE 15-5: GPR-GPR TRANSACTIONS WITH HARDWARE TRIGGERS, SSTP = 1**



## 15.9.2 DESTINATION STOP

When the Destination Stop bit is set (DSTP = 1) and the DMAxDCNT register reloads, the DMA clears the SIRQEN bit to stop receiving new start interrupt request signals and sets the DMAxDCNTIF flag.

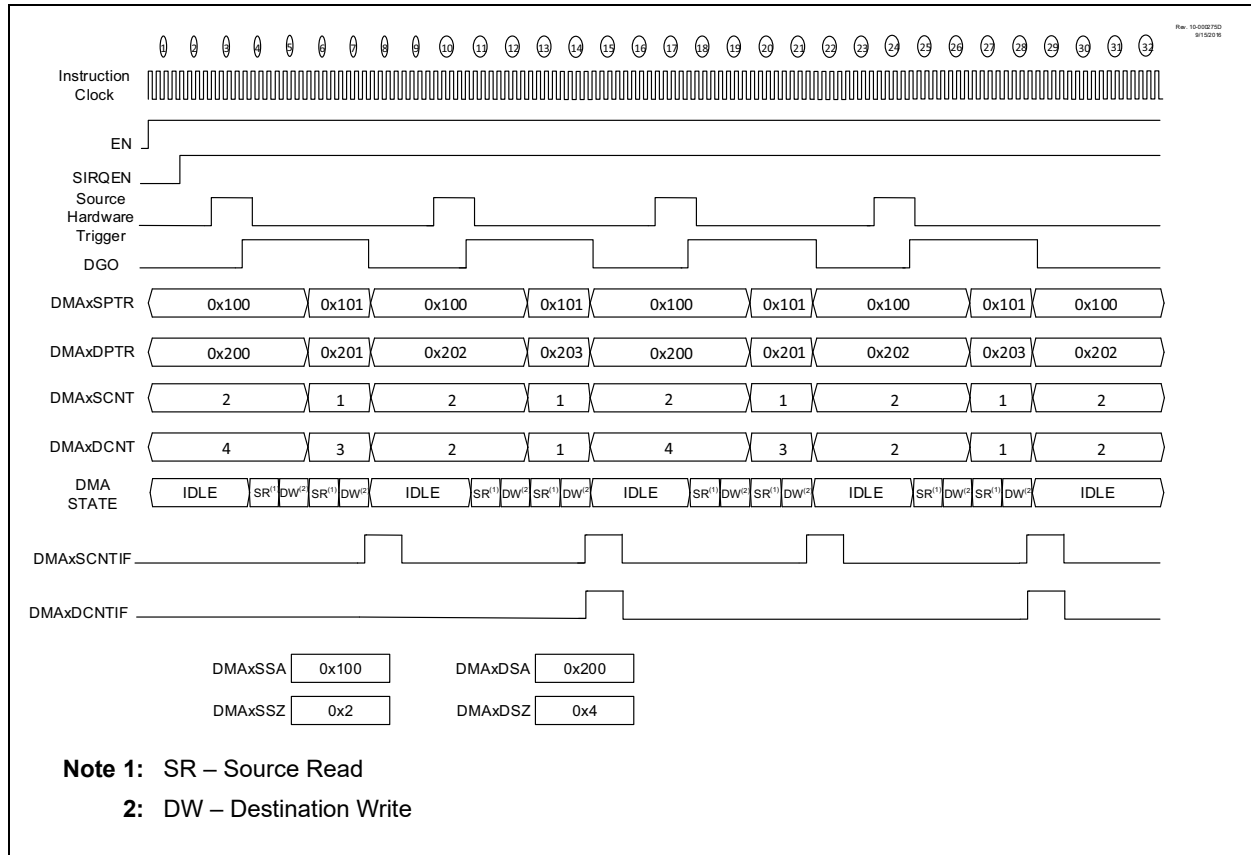
**FIGURE 15-6: GPR-GPR TRANSACTIONS WITH HARDWARE TRIGGERS, DSTP = 1**



## 15.9.3 CONTINUOUS TRANSFER

When the Source or the Destination Stop bit is cleared (SSTP, DSTP = 0), the transactions continue unless cleared by the user. The DMAxSCNTIF and DMAxDCNTIF flags are set whenever the respective counter registers are reloaded.

**FIGURE 15-7: GPR-GPR TRANSACTIONS WITH HARDWARE TRIGGERS, SSTP, DSTP = 0**

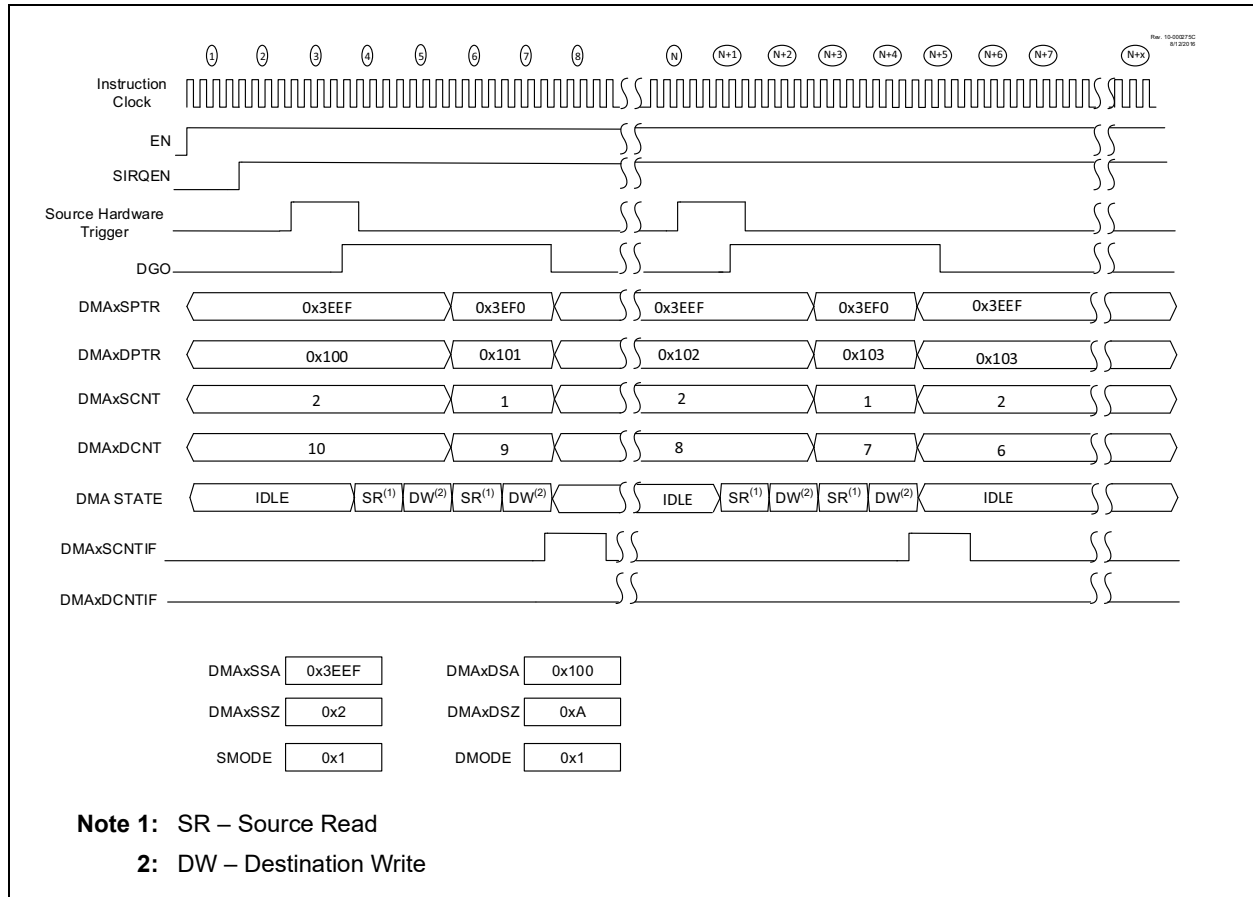


## 15.9.4 TRANSFER FROM SFR TO GPR

The following visual reference describes the sequence of events when copying ADC results to a GPR location. The ADC Interrupt Flag can be chosen as the Source

Hardware trigger, the Source address can be set to point to the ADC Result registers at 3EEF, the Destination address can be set to point to any GPR location of our choice (Example 0x100).

**FIGURE 15-8: SFR SPACE TO GPR SPACE TRANSFER**

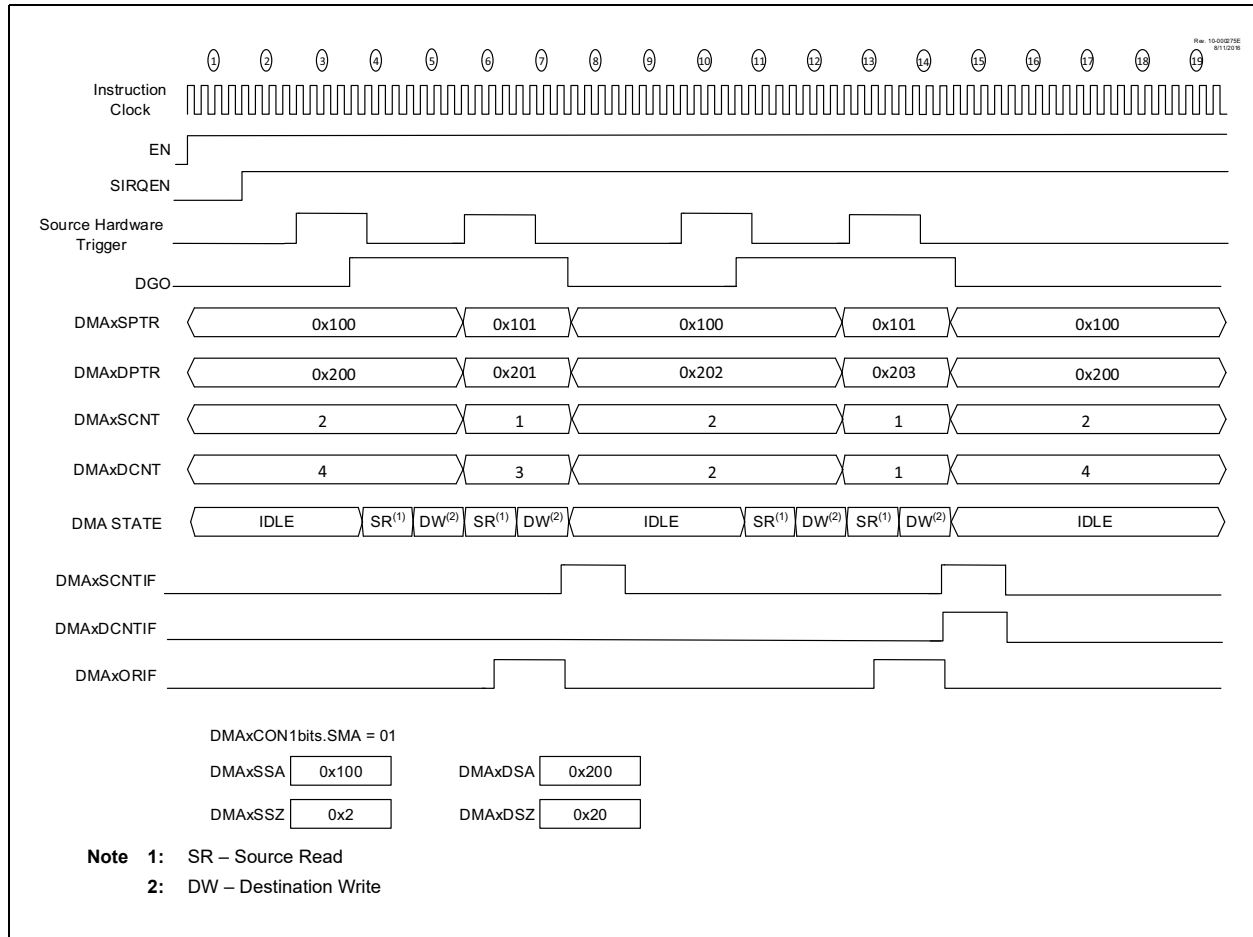




## 15.9.5 OVERRUN INTERRUPT

The Overrun Interrupt flag is set if the DMA receives a trigger to start a new message before the current message is completed.

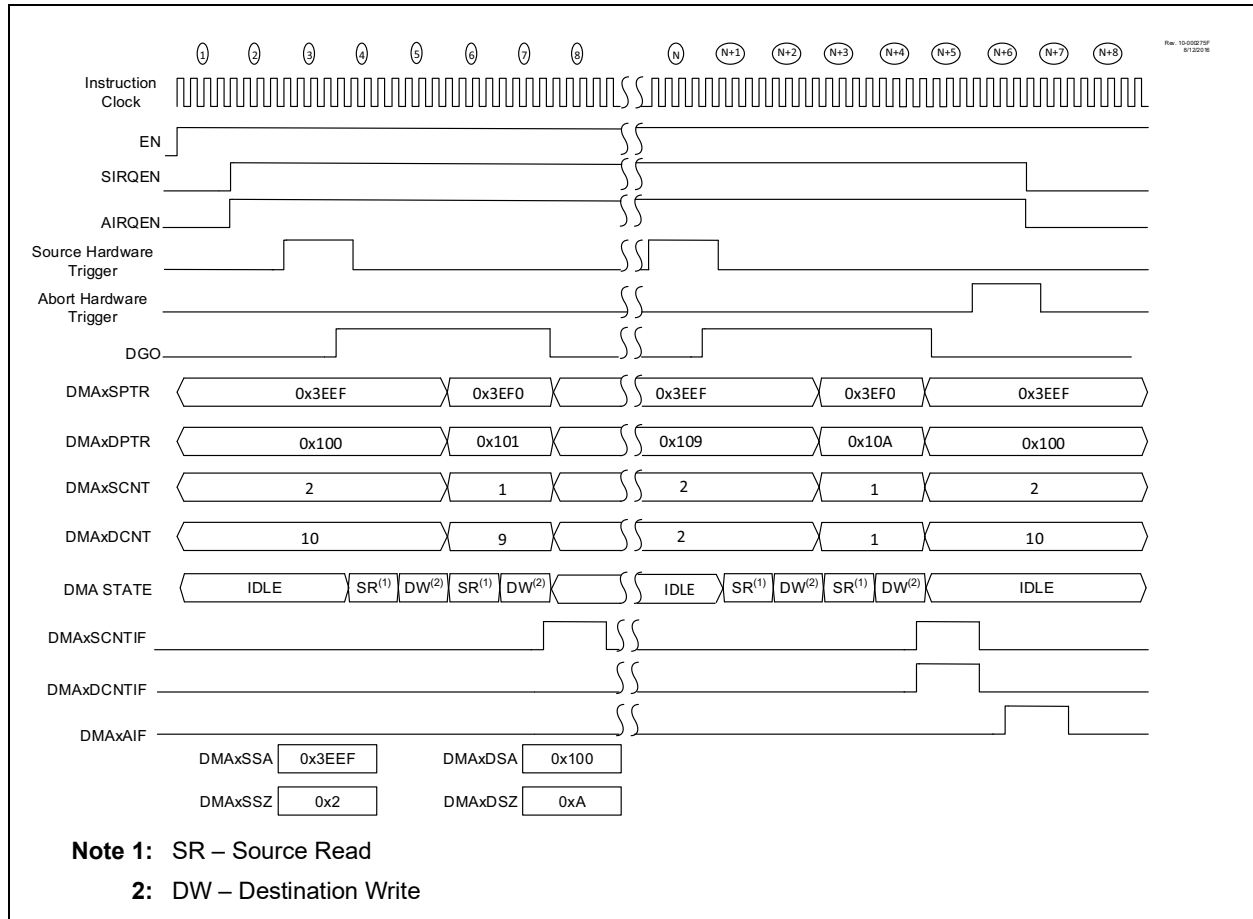
**FIGURE 15-9: OVERRUN INTERRUPT**



## 15.9.6 ABORT TRIGGER, MESSAGE COMPLETE

The AIRQEN needs to be set in order for the DMA to sample Abort Interrupt sources. When an abort interrupt is received the SIRQEN bit is cleared and the AIRQEN bit is cleared to avoid receiving further abort triggers.

**FIGURE 15-10: ABORT AT THE END OF MESSAGE**



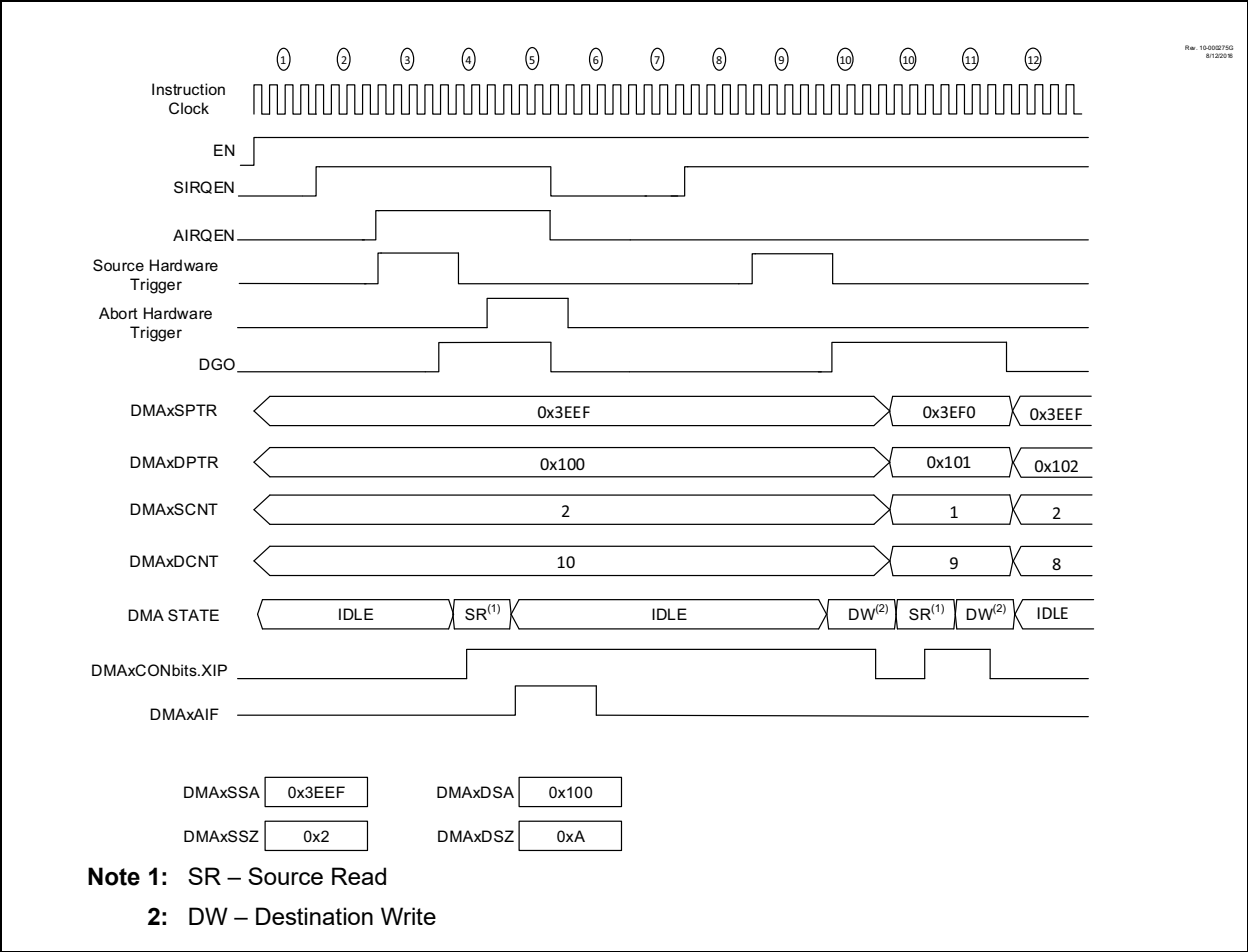
15.9.7 ABORT TRIGGER, MESSAGE IN PROGRESS

When an abort interrupt request is received in the between a DMA transaction, the DMA will perform a soft-stop by clearing the DGO (i.e., if the DMA was reading the source register, it will complete the read operation and then clear the DGO bit)

The SIREQEN bit is cleared to prevent any overrun and the AIRQEN bit is cleared to prevent any false aborts.

When the DGO bit is set again the DMA will resume operation from where it left off after the soft-stop.

FIGURE 15-11: ABORT DURING MESSAGE TRANSFER



The following table contains some of the cases in which the DMA module can be configured to.

**TABLE 15-6: EXAMPLE DMA USE CASE TABLE**

| Source Module                        | Source Register(s)   | Destination Module   | Destination Register(s)                | DCHxSIRQ  |                       |
|--------------------------------------|----------------------|----------------------|----------------------------------------|-----------|-----------------------|
| Signal Measurement<br>Timer<br>(SMT) | SMTxCPW[U:H:L]       | GPR                  | GPR[x,y,z]                             | SMTxPWAIF | Store<br>value        |
|                                      | SMTxCPR[U:H:L]       |                      |                                        | SMTxPRAIF | Store                 |
| GPR/SFR/Program<br>Flash/Data EEPROM | MEMORY[x,y]          | TMR0                 | TMR0[H:L]                              | TMR0IF    | Use<br>custo          |
| GPR/SFR/Program<br>Flash/Data EEPROM | MEMORY[x]            | TMR0                 | PR0                                    | ANY       | Upd<br>base           |
| GPR/SFR/Program<br>Flash/Data EEPROM | MEMORY[x,y]          | TMR1                 | TMR1[H:L]                              | TMR1IF    | Use<br>custo          |
| TMR1                                 | TMR1[H:L]            | GPR                  | GPR[x,y]                               | TMR1GIF   | Use<br>to re<br>regis |
| GPR/SFR/Program<br>Flash/Data EEPROM | MEMORY[x]            | TMR2                 | PR2                                    | TMR2IF    |                       |
| GPR/SFR/Program<br>Flash/Data EEPROM | MEMORY[x,y,z]        | TMR2<br>CCP or PWM   | PR2<br>CCPR[H:L] or<br>PWMDC[H:L]      | ANY       | Freq<br>duty          |
| CCP                                  | CCPR[H:L]            | GPR                  | GPR[x,y]                               | CCPxIF    | Mov<br>Cap            |
| GPR/SFR/Program<br>Flash/Data EEPROM | MEMORY[x,y]          | CCP                  | CCPR[H:L]                              | ANY       | Load<br>valu          |
| GPR/SFR/Program<br>Flash/Data EEPROM | MEMORY [x,y,z,u,v,w] | CCPx<br>CCPy<br>CCPz | CCPxR[H:L]<br>CCPyR[H:L]<br>CCPzR[H:L] | ANY       | Upd<br>at th<br>(e.g. |
| GPR/SFR/Program<br>Flash/Data EEPROM | MEMORY[x,y,z]        | NCO                  | NCOxINC[U:H:L]                         | ANY       | Freq<br>table         |
| GPR/SFR/Program<br>Flash/Data EEPROM | MEMORY[x]            | DAC                  | DACxCON0                               | ANY       | Upd                   |
| GPR/SFR/Program<br>Flash/Data EEPROM | MEMORY[x]            | OSCTUNE              | OSCTUNE                                | ANY       | Auto<br>dithe         |

## 15.10 Reset

The DMA registers are set to the default state on any Reset. The registers are also reset to the default state when the enable bit is cleared (DMA1CON1bits.EN=0).

## 15.11 Power Saving Mode Operation

The DMA utilizes system clocks and it is treated as a peripheral when it comes to power-saving operations. Like other peripherals, the DMA also uses Peripheral Module Disable bits to further tailor its operation in low-power states.

### 15.11.1 SLEEP MODE

When the device enters Sleep mode, the system clock to the module is shut down, therefore no DMA operation is supported in Sleep. Once the system clock is disabled, the requisite read and write clocks are also disabled without which the DMA cannot perform any of its tasks.

Any transfers that may be in progress are resumed on exiting from Sleep mode. Register contents are not affected by the device entering or leaving Sleep mode. It is recommended that DMA transactions be allowed to finish before entering Sleep mode.

### 15.11.2 IDLE MODE

In IDLE mode, all of the system clocks (including the read and write clocks) are still operating but the CPU is not using them to save power.

Therefore, every instruction cycle is available to the system arbiter and if the bubble is granted to the DMA, it may be utilized to move data.

### 15.11.3 DOZE MODE

Similar to the Idle mode, the CPU does not utilize all of the available instruction cycles slots that are available to it in order to save power. It only executes instructions based on its settings from the Doze settings.

Therefore, every instruction not used by the CPU is available for system arbitration and may be utilized by the DMA if granted by the arbiter.

### 15.11.4 PERIPHERAL MODULE DISABLE

The Peripheral Module Disable (PMD) registers provide a method to disable DMA by gating all clock sources supplied to it. The respective DMAxMD bit needs to be set in order to disable the DMA.

## 15.12 DMA Register Interfaces

The DMA can transfer data to any GPR or SFR location. For better user accessibility some of the more commonly used SFR spaces have their Mirror registers placed in a separate data memory location (0x4000-0x40FF), these Mirror registers can be only accessed through the DMA Source and Destination Address registers. Refer to [Table 4-3](#) for details about these mirror registers.

## EXAMPLE 15-1: SETUP DMA1 TO MOVE DATA FROM PROGRAM FLASH MEMORY TO UART1 TRANSMIT BUFFER USING HARDWARE TRIGGERS

```
//This code example illustrates using DMA1 to transfer
//10 bytes of data from 0x1000 in PFM to U1TXB 0x3DEA

void main() {
    //System Initialize
    initializeSystem();

    //Setup UART1
    initializeUART1();

    //Setup DMA1
    //DMA1CON1 - DPTR remains, Source Memory Region PFM, SPTR increments, SSTP
    DMA1CON1 = 0x0B;

    //Source registers
    //Source size
    DMA1SSZH = 0x00;
    DMA1SSZL = 0x0A;

    //Source start address, 0x1000
    DMA1SSAU = 0x00;
    DMA1SSAH = 0x10;
    DMA1SSAL = 0x00;

    //Destination registers
    //Destination size
    DMA1DSZH = 0x00;
    DMA1DSZL = 0x01;

    //Destination start address, 0x3DEA
    DMA1DSAH = 0x3D;
    DMA1DSAL = 0xEA;

    //Start trigger source U1TX
    DMA1SIRQ = 0x1C;

    //Enable & Start DMA transfer
    DMA1CON0 = 0xC0;

    while (1) {
        doSomething();
    }
}
```

### 15.13 Register definitions: DMA

Long bit name prefixes for the DMA peripherals are shown in [Table 15-7](#). Refer to [Section 1.3 “Register and Bit naming conventions”](#) for more information.

**TABLE 15-7: REGISTER AND BIT NAMING**

| Peripheral | Bit Name Prefix |
|------------|-----------------|
| DMA 1      | DMA1            |
| DMA 2      | DMA2            |

## REGISTER 15-1: DMAxCON0: DMAx CONTROL REGISTER 0

| R/W-0/0 | R/W/HC-0/0 | R/W/HS/HC-0/0 | U-0 | U-0 | R/W/HC-0/0 | U-0 | R/HS/HC-0/0 |
|---------|------------|---------------|-----|-----|------------|-----|-------------|
| EN      | SIRQEN     | DGO           | —   | —   | AIRQEN     | —   | XIP         |
| bit 7   |            |               |     |     |            |     | bit 0       |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n/n = Value at POR  
and BOR/Value at all  
other Resets

0 = bit is cleared

x = bit is unknown

u = bit is unchanged

bit 7 **EN:** DMA Module Enable bit

1 = Enables module

0 = Disables module

bit 6 **SIRQEN:** Start of Transfer Interrupt Request Enable bits

1 = Hardware triggers are allowed to start DMA transfers

0 = Hardware triggers are not allowed to start DMA transfers

bit 5 **DGO:** DMA transaction bit

1 = DMA transaction is in progress

0 = DMA transaction is not in progress

bit 4-3 **Unimplemented:** Read as '0'

bit 2 **AIRQEN:** Abort of Transfer Interrupt Request Enable bits

1 = Hardware triggers are allowed to abort DMA transfers

0 = Hardware triggers are not allowed to abort DMA transfers

bit 1 **Unimplemented:** Read as '0'

bit 0 **XIP:** Transfer in Progress Status bit

1 = The DMAxBUF register currently holds contents from a read operation and has not transferred data to the destination.

0 = The DMAxBUF register is empty or has successfully transferred data to the destination address

## REGISTER 15-2: DMAxCON1: DMAx CONTROL REGISTER1

|            |         |         |          |         |            |         |         |
|------------|---------|---------|----------|---------|------------|---------|---------|
| R/W-0/0    | R/W-0/0 | R/W-0/0 | R/W-0/0  | R/W-0/0 | R/W-0/0    | R/W-0/0 | R/W-0/0 |
| DMODE<1:0> |         | DSTP    | SMR<1:0> |         | SMODE<1:0> |         | SSTP    |
| bit 7      |         |         |          |         |            |         | bit 0   |

### Legend:

|                      |                    |                                                       |
|----------------------|--------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit   | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |

- bit 7-6 **DMODE<1:0>**: Destination Address Mode Selection bits
- 11 = Reserved, Do not use
  - 10 = DMAxDPTR<15:0> is decremented after each transfer completion
  - 01 = DMAxDPTR<15:0> is incremented after each transfer completion
  - 00 = DMAxDPTR<15:0> remains unchanged after each transfer completion
- bit 5 **DSTP**: Destination Counter Reload Stop bit
- 1 = SIRQEN bit is cleared when Destination Counter reloads
  - 0 = SIRQEN bit is not cleared when Destination Counter reloads
- bit 4-3 **SMR[1:0]**: Source Memory Region Select bits
- 1x = DMAxSSA<21:0> points to Data EEPROM
  - 01 = DMAxSSA<21:0> points to Program Flash Memory
  - 00 = DMAxSSA<21:0> points to SFR/GPR Data Space
- bit 2-1 **SMODE[1:0]**: Source Address Mode Selection bits
- 11 = Reserved, Do not use
  - 10 = DMAxSPTR<21:0> is decremented after each transfer completion
  - 01 = DMAxSPTR<21:0> is incremented after each transfer completion
  - 00 = DMAxSPTR<21:0> remains unchanged after each transfer completion
- bit 0 **SSTP**: Source Counter Reload Stop bit
- 1 = SIRQEN bit is cleared when Source Counter reloads
  - 0 = SIRQEN bit is not cleared when Source Counter reloads



## REGISTER 15-3: DMAxBUF: DMAx DATA BUFFER REGISTER

|       |      |      |      |      |      |      |       |
|-------|------|------|------|------|------|------|-------|
| R-0   | R-0  | R-0  | R-0  | R-0  | R-0  | R-0  | R-0   |
| BUF7  | BUF6 | BUF5 | BUF4 | BUF3 | BUF2 | BUF1 | BUF0  |
| bit 7 |      |      |      |      |      |      | bit 0 |

### Legend:

R = Readable bit      W = Writable bit      U = Unimplemented bit, read as '0'  
 -n/n = Value at POR      1 = bit is set      0 = bit is cleared      x = bit is unknown  
 and BOR/Value at all      u = bit is unchanged  
 other Resets

bit 7-0 **BUF<7:0>**: DMA Internal Data Buffer bits

DMABUF<7:0>

These bits reflect the content of the internal data buffer the DMA peripheral uses to hold the data being moved from the source to destination.

## REGISTER 15-4: DMAxSSAL: DMAx SOURCE START ADDRESS LOW REGISTER

|          |         |         |         |         |         |         |         |
|----------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0  | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| SSA<7:0> |         |         |         |         |         |         |         |
| bit 7    |         |         |         |         |         |         | bit 0   |

### Legend:

R = Readable bit      W = Writable bit      U = Unimplemented bit, read as '0'  
 -n/n = Value at POR      1 = bit is set      0 = bit is cleared      x = bit is unknown  
 and BOR/Value at all      u = bit is unchanged  
 other Resets

bit 7-0 **SSA<7:0>**: Source Start Address bits

## REGISTER 15-5: DMAxSSAH: DMAx SOURCE START ADDRESS HIGH REGISTER

|           |         |         |         |         |         |         |         |
|-----------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0   | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| SSA<15:8> |         |         |         |         |         |         |         |
| bit 7     |         |         |         |         |         |         | bit 0   |

### Legend:

R = Readable bit      W = Writable bit      U = Unimplemented bit, read as '0'  
 -n/n = Value at POR and      1 = bit is set      0 = bit is cleared      x = bit is unknown  
 BOR/Value at all other      u = bit is unchanged  
 Resets

bit 7-0 **SSA<15:8>**: Source Start Address bits

## REGISTER 15-6: DMAxSSAU: DMAx SOURCE START ADDRESS UPPER REGISTER

|       |     |            |         |         |         |         |         |
|-------|-----|------------|---------|---------|---------|---------|---------|
| U-0   | U-0 | R/W-0/0    | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| —     | —   | SSA<21:16> |         |         |         |         |         |
| bit 7 |     |            |         |         |         |         | bit 0   |

### Legend:

R = Readable bit      W = Writable bit      U = Unimplemented bit, read as '0'  
 -n/n = Value at POR      1 = bit is set      0 = bit is cleared      x = bit is unknown  
 and BOR/Value at all      u = bit is unchanged  
 other Resets

bit 7-0      **SSA<21:16>**: Source Start Address bits

## REGISTER 15-7: DMAxSPTRL: DMAx SOURCE POINTER LOW REGISTER

|           |     |     |     |     |     |     |       |
|-----------|-----|-----|-----|-----|-----|-----|-------|
| R-0       | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0   |
| SPTR<7:0> |     |     |     |     |     |     |       |
| bit 7     |     |     |     |     |     |     | bit 0 |

### Legend:

R = Readable bit      W = Writable bit      U = Unimplemented bit, read as '0'  
 -n/n = Value at POR and      1 = bit is set      0 = bit is cleared      x = bit is unknown  
 BOR/Value at all other      u = bit is unchanged  
 Resets

bit 15-0      **SPTR<7:0>**: Current Source Address Pointer

## REGISTER 15-8: DMAxSPTRH: DMAx SOURCE POINTER HIGH REGISTER

|            |     |     |     |     |     |     |       |
|------------|-----|-----|-----|-----|-----|-----|-------|
| R-0        | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0   |
| SPTR<15:8> |     |     |     |     |     |     |       |
| bit 7      |     |     |     |     |     |     | bit 0 |

### Legend:

R = Readable bit      W = Writable bit      U = Unimplemented bit, read as '0'  
 -n/n = Value at POR and      1 = bit is set      0 = bit is cleared      x = bit is unknown  
 BOR/Value at all other      u = bit is unchanged  
 Resets

bit 5-0      **SPTR<15:8>**: Current Source Address Pointer

## REGISTER 15-9: DMAxSPTRU: DMAx SOURCE POINTER UPPER REGISTER

|       |     |             |     |     |     |     |       |
|-------|-----|-------------|-----|-----|-----|-----|-------|
| U-0   | U-0 | R-0         | R-0 | R-0 | R-0 | R-0 | R-0   |
| —     | —   | SPTR<21:16> |     |     |     |     |       |
| bit 7 |     |             |     |     |     |     | bit 0 |

### Legend:

R = Readable bit      W = Writable bit      U = Unimplemented bit, read as '0'  
 -n/n = Value at POR and      1 = bit is set      0 = bit is cleared      x = bit is unknown  
 BOR/Value at all other      u = bit is unchanged  
 Resets

bit 7-6      **Unimplemented:** Read as '0'  
 bit 5-0      **SPTR<21:16>:** Current Source Address Pointer

## REGISTER 15-10: DMAxSSZL: DMAx SOURCE SIZE LOW REGISTER

|          |         |         |         |         |         |         |         |
|----------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0  | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| SSZ<7:0> |         |         |         |         |         |         |         |
| bit 7    |         |         |         |         |         |         | bit 0   |

### Legend:

R = Readable bit      W = Writable bit      U = Unimplemented bit, read as '0'  
 -n/n = Value at POR and      1 = bit is set      0 = bit is cleared      x = bit is unknown  
 BOR/Value at all other      u = bit is unchanged  
 Resets

bit 7-0      **SSZ<7:0>:** Source Message Size bits

## REGISTER 15-11: DMAxSSZH: DMAx SOURCE SIZE HIGH REGISTER

|       |     |     |     |           |         |         |         |
|-------|-----|-----|-----|-----------|---------|---------|---------|
| U-0   | U-0 | U-0 | U-0 | R/W-0/0   | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| —     | —   | —   | —   | SSZ<11:8> |         |         |         |
| bit 7 |     |     |     |           |         |         | bit 0   |

### Legend:

R = Readable bit      W = Writable bit      U = Unimplemented bit, read as '0'  
 -n/n = Value at POR and      1 = bit is set      0 = bit is cleared      x = bit is unknown  
 BOR/Value at all other      u = bit is unchanged  
 Resets

bit 7-4      **Unimplemented:** Read as '0'  
 bit 3-0      **SSZ<11:8>:** Source Message Size bits

## REGISTER 15-12: DMAxSCNTL: DMAx SOURCE COUNT LOW REGISTER

|           |     |     |     |     |     |     |       |
|-----------|-----|-----|-----|-----|-----|-----|-------|
| R-0       | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0   |
| SCNT<7:0> |     |     |     |     |     |     |       |
| bit 7     |     |     |     |     |     |     | bit 0 |

### Legend:

R = Readable bit      W = Writable bit      U = Unimplemented bit, read as '0'  
 -n/n = Value at POR and BOR/Value at all other Resets      1 = bit is set      0 = bit is cleared      x = bit is unknown      u = bit is unchanged

bit 7-0      **SCNT<7:0>**: Current Source Byte Count

## REGISTER 15-13: DMAxSCNTH: DMAx SOURCE COUNT HIGH REGISTER

|       |     |     |     |            |     |     |       |
|-------|-----|-----|-----|------------|-----|-----|-------|
| U-0   | U-0 | U-0 | U-0 | R-0        | R-0 | R-0 | R-0   |
| —     | —   | —   | —   | SCNT<11:8> |     |     |       |
| bit 7 |     |     |     |            |     |     | bit 0 |

### Legend:

R = Readable bit      W = Writable bit      U = Unimplemented bit, read as '0'  
 -n/n = Value at POR and BOR/Value at all other Resets      1 = bit is set      0 = bit is cleared      x = bit is unknown      u = bit is unchanged

bit 7-4      **Unimplemented**: Read as '0'

bit 3-0      **SCNT<11:8>**: Current Source Byte Count

## REGISTER 15-14: DMAxDSAL: DMAx DESTINATION START ADDRESS LOW REGISTER

|          |         |         |         |         |         |         |         |
|----------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0  | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| DSA<7:0> |         |         |         |         |         |         |         |
| bit 7    |         |         |         |         |         |         | bit 0   |

### Legend:

R = Readable bit      W = Writable bit      U = Unimplemented bit, read as '0'  
 -n/n = Value at POR and BOR/Value at all other Resets      1 = bit is set      0 = bit is cleared      x = bit is unknown      u = bit is unchanged

bit 7-0      **DSA<7:0>**: Destination Start Address bits

## REGISTER 15-15: DMAxDSAH: DMAx DESTINATION START ADDRESS HIGH REGISTER

|           |         |         |         |         |         |         |         |
|-----------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0   | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| DSA<15:8> |         |         |         |         |         |         |         |
| bit 7     |         |         |         |         |         |         | bit 0   |

### Legend:

R = Readable bit      W = Writable bit      U = Unimplemented bit, read as '0'  
 -n/n = Value at POR and      1 = bit is set      0 = bit is cleared      x = bit is unknown  
 BOR/Value at all other                u = bit is unchanged  
 Resets

bit 7-0      **DSA<15:8>**: Destination Start Address bits

## REGISTER 15-16: DMAxDPTRL: DMAx DESTINATION POINTER LOW REGISTER

|           |     |     |     |     |     |     |       |
|-----------|-----|-----|-----|-----|-----|-----|-------|
| R-0       | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0   |
| DPTR<7:0> |     |     |     |     |     |     |       |
| bit 7     |     |     |     |     |     |     | bit 0 |

### Legend:

R = Readable bit      W = Writable bit      U = Unimplemented bit, read as '0'  
 -n/n = Value at POR and      1 = bit is set      0 = bit is cleared      x = bit is unknown  
 BOR/Value at all other                u = bit is unchanged  
 Resets

bit 7-0      **DPTR<7:0>**: Current Destination Address Pointer

## REGISTER 15-17: DMAxDPTRH: DMAx DESTINATION POINTER HIGH REGISTER

|            |     |     |     |     |     |     |       |
|------------|-----|-----|-----|-----|-----|-----|-------|
| R-0        | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0   |
| DPTR<15:8> |     |     |     |     |     |     |       |
| bit 7      |     |     |     |     |     |     | bit 0 |

### Legend:

R = Readable bit      W = Writable bit      U = Unimplemented bit, read as '0'  
 -n/n = Value at POR and      1 = bit is set      0 = bit is cleared      x = bit is unknown  
 BOR/Value at all other                u = bit is unchanged  
 Resets

bit 7-0      **DPTR<15:8>**: Current Destination Address Pointer

## REGISTER 15-18: DMAxDSZL: DMAx DESTINATION SIZE LOW REGISTER

|          |         |         |         |         |         |         |         |
|----------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0  | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| DSZ<7:0> |         |         |         |         |         |         |         |
| bit 7    |         |         |         |         |         |         | bit 0   |

### Legend:

R = Readable bit      W = Writable bit      U = Unimplemented bit, read as '0'  
 -n/n = Value at POR and      1 = bit is set      0 = bit is cleared      x = bit is unknown  
 BOR/Value at all other                u = bit is unchanged  
 Resets

bit 7-0      **DSZ<7:0>**: Destination Message Size bits

## REGISTER 15-19: DMAxDSZH: DMAx DESTINATION SIZE HIGH REGISTER

|       |     |     |     |           |         |         |         |
|-------|-----|-----|-----|-----------|---------|---------|---------|
| U-0   | U-0 | U-0 | U-0 | R/W-0/0   | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| —     | —   | —   | —   | DSZ<11:8> |         |         |         |
| bit 7 |     |     |     |           |         |         | bit 0   |

### Legend:

R = Readable bit      W = Writable bit      U = Unimplemented bit, read as '0'  
 -n/n = Value at POR and      1 = bit is set      0 = bit is cleared      x = bit is unknown  
 BOR/Value at all other                u = bit is unchanged  
 Resets

bit 7-4      **Unimplemented**: Read as '0'  
 bit 3-0      **DSZ<11:8>**: Destination Message Size bits

## REGISTER 15-20: DMAxDCNTL: DMAx DESTINATION COUNT LOW REGISTER

|           |     |     |     |     |     |     |       |
|-----------|-----|-----|-----|-----|-----|-----|-------|
| R-0       | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0   |
| DCNT<7:0> |     |     |     |     |     |     |       |
| bit 7     |     |     |     |     |     |     | bit 0 |

### Legend:

R = Readable bit      W = Writable bit      U = Unimplemented bit, read as '0'  
 -n/n = Value at POR and      1 = bit is set      0 = bit is cleared      x = bit is unknown  
 BOR/Value at all other                u = bit is unchanged  
 Resets

bit 7-0      **DCNT<7:0>**: Current Destination Byte Count



**TABLE 15-8: DMAxSIRQ AND DMAxAIRQ TRIGGER SOURCES**

| DMAxSIRQ<br>DMAxAIRQ | Trigger<br>Source <sup>(2)</sup> | Level<br>Triggered <sup>(1)</sup> | DMAxSIRQ<br>DMAxAIRQ | Trigger<br>Source | Level<br>Triggered |
|----------------------|----------------------------------|-----------------------------------|----------------------|-------------------|--------------------|
| 0x0                  | Reserved                         |                                   | 0x2A                 | TXB0IF            | No                 |
| 0x1                  | HLVDIF                           | No                                | 0x2B                 | TXB1IF            | No                 |
| 0x2                  | OSFIF                            | No                                | 0x2C                 | TXB2IF/TXBnIF     | No                 |
| 0x3                  | CSWIF                            | No                                | 0x2D                 | ERRIF             | No                 |
| 0x4                  | NVMIF                            | No                                | 0x2E                 | WAKIF             | No                 |
| 0x5                  | SCANIF                           | No                                | 0x2F                 | IRXIF             | No                 |
| 0x6                  | CRCIF                            | No                                | 0x30                 | CMP2IF            | No                 |
| 0x7                  | IOCIF                            | Yes                               | 0x31                 | SMT2IF            | No                 |
| 0x8                  | INT0IF                           | No                                | 0x32                 | SMT2PRAIF         | No                 |
| 0x9                  | ZCDIF                            | No                                | 0x33                 | SMT2PWAIF         | No                 |
| 0xA                  | ADIF                             | No                                | 0x34                 | DMA2SCNTIF        | No                 |
| 0xB                  | ADTIF                            | No                                | 0x35                 | DMA2DCNTIF        | No                 |
| 0xC                  | CMP1IF                           | No                                | 0x36                 | DMA2ORIF          | No                 |
| 0xD                  | SMT1IF                           | No                                | 0x37                 | DMA2AIF           | No                 |
| 0xE                  | SMT1PRAIF                        | No                                | 0x38                 | I2C2RXIF          | Yes                |
| 0xF                  | SMT1PWAIF                        | No                                | 0x39                 | I2C2TXIF          | Yes                |
| 0x10                 | DMA1SCNTIF                       | No                                | 0x3A                 | I2C2IF            | Yes                |
| 0x11                 | DMA1DCNTIF                       | No                                | 0x3B                 | I2C2EIF           | Yes                |
| 0x12                 | DMA1ORIF                         | No                                | 0x3C                 | U2RXIF            | Yes                |
| 0x13                 | DMA1AIF                          | No                                | 0x3D                 | U2TXIF            | Yes                |
| 0x14                 | SPI1RXIF                         | Yes                               | 0x3E                 | U2EIF             | Yes                |
| 0x15                 | SPI1TXIF                         | Yes                               | 0x3F                 | U2IF              | Yes                |
| 0x16                 | SPI1IF                           | Yes                               | 0x40                 | TMR3IF            | No                 |
| 0x17                 | I2C1RXIF                         | Yes                               | 0x41                 | TMR3GIF           | No                 |
| 0x18                 | I2C1TXIF                         | Yes                               | 0x42                 | TMR4IF            | No                 |
| 0x19                 | I2C1IF                           | Yes                               | 0x43                 | CCP2IF            | No                 |
| 0x1A                 | I2C1EIF                          | Yes                               | 0x44                 | CWG2IF            | No                 |
| 0x1B                 | U1RXIF                           | Yes                               | 0x45                 | CLC2IF            | No                 |
| 0x1C                 | U1TXIF                           | Yes                               | 0x46                 | INT2IF            | No                 |
| 0x1D                 | U1EIF                            | Yes                               | 0x47                 | TMR5IF            | No                 |
| 0x1E                 | U1IF                             | No                                | 0x48                 | TMR5GIF           | No                 |
| 0x1F                 | TMR0IF                           | No                                | 0x49                 | TMR6IF            | No                 |
| 0x20                 | TMR1IF                           | No                                | 0x4A                 | CCP3IF            | No                 |
| 0x21                 | TMR1GIF                          | No                                | 0x4B                 | CWG3IF            | No                 |
| 0x22                 | TMR2IF                           | No                                | 0x4C                 | CLC3IF            | No                 |
| 0x23                 | CCP1IF                           | No                                | 0x4D                 | CCP4IF            | No                 |
| 0x24                 | NCOIF                            | Yes                               | 0x4E                 | CLC4IF            | No                 |
| 0x25                 | CWG1IF                           | No                                | 0x4F                 | Reserved          |                    |
| 0x26                 | CLC1IF                           | No                                | —                    |                   |                    |
| 0x27                 | INT1IF                           | No                                | 0xFF                 |                   |                    |
| 0x28                 | RXB0IF/FIF0IF                    | No                                |                      |                   |                    |
| 0x29                 | RXB1IF/RXBnIF                    | No                                |                      |                   |                    |

**Note 1:** All trigger sources that are not Level-triggered are Edge-triggered.

**2:** The event that sets the flag is the interrupt trigger, not the flag itself. The flag remains set.



**TABLE 15-9: SUMMARY OF REGISTERS ASSOCIATED WITH DMA**

| Name      | Bit 7      | Bit 6     | Bit 5       | Bit 4    | Bit 3      | Bit 2      | Bit 1 | Bit 0 | Register on Page    |
|-----------|------------|-----------|-------------|----------|------------|------------|-------|-------|---------------------|
| DMAxCON0  | EN         | SIRQEN    | DGO         | —        | —          | AIRQEN     | —     | XIP   | <a href="#">239</a> |
| DMAxCON1  | DMODE<1:0> |           | DSTP        | SMR<1:0> |            | SMODE<1:0> |       | SSTP  | <a href="#">240</a> |
| DMAxBUF   | DBUF7      | DBUF6     | DBUF5       | DBUF4    | DBUF3      | DBUF2      | DBUF1 | DBUF0 | <a href="#">241</a> |
| DMAxSSAL  | SSA<7:0>   |           |             |          |            |            |       |       | <a href="#">241</a> |
| DMAxSSAH  | SSA<15:8>  |           |             |          |            |            |       |       | <a href="#">241</a> |
| DMAxSSAU  | —          | —         | SSA<21:16>  |          |            |            |       |       | <a href="#">242</a> |
| DMAxSPTRL | SPTR<7:0>  |           |             |          |            |            |       |       | <a href="#">242</a> |
| DMAxSPTRH | SPTR<15:8> |           |             |          |            |            |       |       | <a href="#">242</a> |
| DMAxSPTRU | —          | —         | SPTR<21:16> |          |            |            |       |       | <a href="#">243</a> |
| DMAxSSZL  | SSZ<7:0>   |           |             |          |            |            |       |       | <a href="#">243</a> |
| DMAxSSZH  | —          | —         | —           | —        | SSZ<11:8>  |            |       |       | <a href="#">243</a> |
| DMAxSCNTL | SCNT<7:0>  |           |             |          |            |            |       |       | <a href="#">244</a> |
| DMAxSCNTH | —          | —         | —           | —        | SCNT<11:8> |            |       |       | <a href="#">244</a> |
| DMAxDSAL  | DSA<7:0>   |           |             |          |            |            |       |       | <a href="#">244</a> |
| DMAxDSAH  | DSA<15:8>  |           |             |          |            |            |       |       | <a href="#">245</a> |
| DMAxDPTRL | DPTR<7:0>  |           |             |          |            |            |       |       | <a href="#">245</a> |
| DMAxDPTRH | DPTR<15:8> |           |             |          |            |            |       |       | <a href="#">245</a> |
| DMAxDSZL  | DSZ<7:0>   |           |             |          |            |            |       |       | <a href="#">246</a> |
| DMAxDSZH  | —          | —         | —           | —        | DSZ<11:8>  |            |       |       | <a href="#">246</a> |
| DMAxDCNTL | DCNT<7:0>  |           |             |          |            |            |       |       | <a href="#">246</a> |
| DMAxDCNTH | —          | —         | —           | —        | DCNT<11:8> |            |       |       | <a href="#">247</a> |
| DMAxSIRQ  | —          | SIRQ<6:0> |             |          |            |            |       |       | <a href="#">247</a> |
| DMAxAIRQ  | —          | AIRQ<6:0> |             |          |            |            |       |       | <a href="#">247</a> |

**Legend:** — = unimplemented location, read as '0'. Shaded cells are not used by DMA.

## 16.0 I/O PORTS

The PIC18(L)F25/26K83 devices have four I/O ports, allocated as shown in [Table 16-1](#).

**TABLE 16-1: PORT ALLOCATION TABLE FOR PIC18(L)F25/26K83 DEVICES**

| Device         | PORTA | PORTB | PORTC | PORTE            |
|----------------|-------|-------|-------|------------------|
| PIC18(L)F25K83 | •     | •     | •     | • <sup>(1)</sup> |
| PIC18(L)F26K83 | •     | •     | •     | • <sup>(1)</sup> |

**Note 1:** Pin RE3 only.

Each port has ten registers to control the operation. These registers are:

- PORTx registers (reads the levels on the pins of the device)
- LATx registers (output latch)
- TRISx registers (data direction)
- ANSELx registers (analog select)
- WPUx registers (weak pull-up)
- INLVx registers (input level control)
- SLRCONx registers (slew rate control)
- ODCONx registers (open-drain control)

Outside of registers to control bits of all the ports, the two following registers are also present:

- RxyI2C (I<sup>2</sup>C pad control)

Most port pins share functions with device peripherals, both analog and digital. In general, when a peripheral is enabled on a port pin, that pin cannot be used as a general purpose output; however, the pin can still be read.

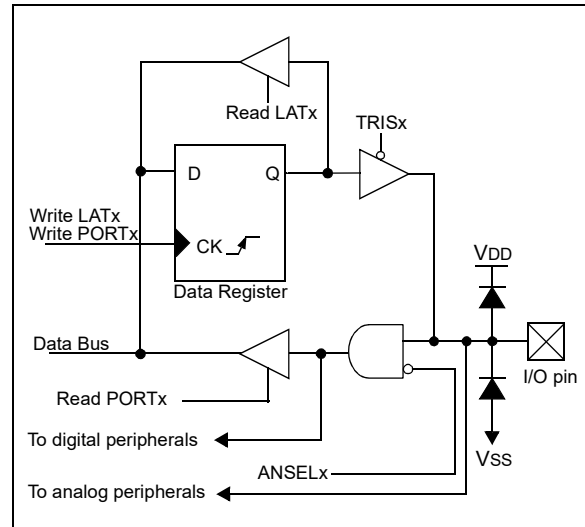
The Data Latch (LATx registers) is useful for read-modify-write operations on the value that the I/O pins are driving.

A write operation to the LATx register has the same effect as a write to the corresponding PORTx register. A read of the LATx register reads of the values held in the I/O PORT latches, while a read of the PORTx register reads the actual I/O pin value.

Ports that support analog inputs have an associated ANSELx register. When an ANSELx bit is set, the digital input buffer associated with that bit is disabled.

Disabling the input buffer prevents analog signal levels on the pin between a logic high and low from causing excessive current in the logic input circuitry. A simplified model of a generic I/O port, without the interfaces to other peripherals, is shown in [Figure 16.1](#).

## 16.1 GENERIC I/O PORT OPERATION



## 16.2 I/O Priorities

Each pin defaults to the PORT data latch after Reset. Other functions are selected with the peripheral pin select logic. See [Section 17.0 “Peripheral Pin Select \(PPS\) Module”](#) for more information.

Analog input functions, such as ADC and comparator inputs, are not shown in the peripheral pin select lists. These inputs are active when the I/O pin is set for Analog mode using the ANSELx register. Digital output functions may continue to control the pin when it is in Analog mode.

Analog outputs, when enabled, take priority over digital outputs and force the digital output driver into a high-impedance state.

The pin function priorities are as follows:

1. Configuration bits
2. Analog outputs (disable the input buffers)
3. Analog inputs
4. Port inputs and outputs from PPS

## 16.3 PORTx Registers

In this section the generic names such as PORTx, LATx, TRISx, etc. can be associated with PORTA, PORTB, and PORTC ports. The functionality of PORTE is different compared to other ports and is explained in a separate section.

## 16.3.1 DATA REGISTER

PORTx is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISx (Register 16-2). Setting a TRISx bit ('1') will make the corresponding PORTx pin an input (i.e., disable the output driver). Clearing a TRISx bit ('0') will make the corresponding PORTx pin an output (i.e., it enables output driver and puts the contents of the output latch on the selected pin). Example 16-1 shows how to initialize PORTx.

Reading the PORTx register (Register 16-1) reads the status of the pins, whereas writing to it will write to the PORT latch. All write operations are read-modify-write operations. Therefore, a write to a port implies that the port pins are read, this value is modified and then written to the PORT data latch (LATx).

The PORT data latch LATx (Register 16-3) holds the output port data and contains the latest value of a LATx or PORTx write.

### EXAMPLE 16-1: INITIALIZING PORTA

```
; This code example illustrates
; initializing the PORTA register. The
; other ports are initialized in the same
; manner.

BANKSEL PORTA      ;
CLRF PORTA         ;Init PORTA
BANKSEL LATA        ;Data Latch
CLRF LATA          ;
BANKSEL ANSELA      ;
CLRF ANSELA         ;digital I/O
BANKSEL TRISA       ;
MOVLW B'11111000'  ;Set RA<7:3> as inputs
MOVWF TRISA         ;and set RA<2:0> as
                   ;outputs
```

## 16.3.2 DIRECTION CONTROL

The TRISx register (Register 16-2) controls the PORTx pin output drivers, even when they are being used as analog inputs. The user should ensure the bits in the TRISx register are maintained set when using them as analog inputs. I/O pins configured as analog inputs always read '0'.

## 16.3.3 ANALOG CONTROL

The ANSELx register (Register 16-4) is used to configure the Input mode of an I/O pin to analog. Setting the appropriate ANSELx bit high will cause all digital reads on the pin to be read as '0' and allow analog functions on the pin to operate correctly.

The state of the ANSELx bits has no effect on digital output functions. A pin with TRIS clear and ANSEL set will still operate as a digital output, but the Input mode will be analog. This can cause unexpected behavior when executing read-modify-write instructions on the affected port.

**Note:** The ANSELx bits default to the Analog mode after Reset. To use any pins as digital general purpose or peripheral inputs, the corresponding ANSEL bits must be initialized to '0' by user software.

## 16.3.4 OPEN-DRAIN CONTROL

The ODCONx register (Register 16-6) controls the open-drain feature of the port. Open-drain operation is independently selected for each pin. When an ODCONx bit is set, the corresponding port output becomes an open-drain driver capable of sinking current only. When an ODCONx bit is cleared, the corresponding port output pin is the standard push-pull drive capable of sourcing and sinking current.

**Note:** It is necessary to set open-drain control when using the pin for I<sup>2</sup>C.

## 16.3.5 SLEW RATE CONTROL

The SLRCONx register (Register 16-7) controls the slew rate option for each port pin. Slew rate for each port pin can be controlled independently. When an SLRCONx bit is set, the corresponding port pin drive is slew rate limited. When an SLRCONx bit is cleared, The corresponding port pin drive slews at the maximum rate possible.

## 16.3.6 INPUT THRESHOLD CONTROL

The INLV<sub>Lx</sub> register ([Register 16-8](#)) controls the input voltage threshold for each of the available PORT<sub>x</sub> input pins. A selection between the Schmitt Trigger CMOS or the TTL compatible thresholds is available. The input threshold is important in determining the value of a read of the PORT<sub>x</sub> register and also the level at which an interrupt-on-change occurs, if that feature is enabled. See [Table 45-4](#) for more information on threshold levels.

**Note:** Changing the input threshold selection should be performed while all peripheral modules are disabled. Changing the threshold level during the time a module is active may inadvertently generate a transition associated with an input pin, regardless of the actual voltage level on that pin.

## 16.3.7 WEAK PULL-UP CONTROL

The WPU<sub>x</sub> register ([Register 16-5](#)) controls the individual weak pull-ups for each port pin.

## 16.3.8 EDGE SELECTABLE INTERRUPT-ON-CHANGE

An interrupt can be generated by detecting a signal at the port pin that has either a rising edge or a falling edge. Any individual pin can be configured to generate an interrupt. The interrupt-on-change module is present on all the pins of Ports B, C, E and on pin RG5. For further details about the IOC module refer to [Section 18.0 “Interrupt-on-Change”](#).

## 16.3.9 I<sup>2</sup>C PAD CONTROL

For the PIC18(L)F25/26K83 devices, the I<sup>2</sup>C specific pads are available on RB1, RB2, RC3, RC4, RD0<sup>(1)</sup> and RD1<sup>(1)</sup> pins. The I<sup>2</sup>C characteristics of each of these pins is controlled by the RxyI2C registers (see [Register 16-9](#)). These characteristics include enabling I<sup>2</sup>C specific slew rate (over standard GPIO slew rate), selecting internal pull-ups for I<sup>2</sup>C pins, and selecting appropriate input threshold as per SMBus specifications.

**Note 1:** RD0 and RD1 I<sup>2</sup>C pads are not available in PIC18(L)F25K83 parts.

**2:** Any peripheral using the I<sup>2</sup>C pins read the I<sup>2</sup>C ST inputs when enabled via RxyI2C.

## 16.4 PORTE Registers

### 16.4.1 MASTER CLEAR INPUT ( $\overline{\text{MCLR}}$ )

For PIC18(L)F2xK83 devices, PORTE is only available when Master Clear functionality is disabled (MCLRE = 0). In this case, PORTE is a single bit, input-only port comprised of RE3 only. The pin operates as previously described. RE3 in PORTE register is a read-only bit and will read ‘1’ when MCLRE = 1 (i.e., Master Clear enabled).

### 16.4.2 RE3 WEAK PULL-UP

The port RE3 pin has an individually controlled weak internal pull-up. When set, the WPUE3 bit enables the RE3 pin pull-up. When the RE3 port pin is configured as  $\overline{\text{MCLR}}$ , (CONFIG2L, MCLRE = 1 and CONFIG4H, LVP = 0), or configured for Low-Voltage Programming, (MCLRE = x and LVP = 1), the pull-up is always enabled and the WPUE3 bit has no effect.

### 16.4.3 INTERRUPT-ON-CHANGE

The interrupt-on-change feature is available only on the RE3 pin of PORTE for all devices. If MCLRE = 1 or LVP = 1, RE3 port functionality is disabled and interrupt-on-change on RE3 is not available. For further details refer to [Section 18.0 “Interrupt-on-Change”](#).

## 16.5 Register Definitions: Port Control

**REGISTER 16-1: PORTx: PORTx REGISTER<sup>(1)</sup>**

| R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
|---------|---------|---------|---------|---------|---------|---------|---------|
| Rx7     | Rx6     | Rx5     | Rx4     | Rx3     | Rx2     | Rx1     | Rx0     |
| bit 7   |         |         |         |         |         |         | bit 0   |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

bit 7-0 **Rx<7:0>**: Rx7:Rx0 Port I/O Value bits

1 = Port pin is  $\geq V_{IH}$

0 = Port pin is  $\leq V_{IL}$

**Note 1:** Writes to PORTx are actually written to the corresponding LATx register.  
Reads from PORTx register return actual I/O pin values.

**TABLE 16-2: PORT REGISTERS**

| Name  | Bit 7              | Bit 6              | Bit 5 | Bit 4 | Bit 3              | Bit 2 | Bit 1 | Bit 0 |
|-------|--------------------|--------------------|-------|-------|--------------------|-------|-------|-------|
| PORTA | RA7                | RA6                | RA5   | RA4   | RA3                | RA2   | RA1   | RA0   |
| PORTB | RB7 <sup>(1)</sup> | RB6 <sup>(1)</sup> | RB5   | RB4   | RB3                | RB2   | RB1   | RB0   |
| PORTC | RC7                | RC6                | RC5   | RC4   | RC3                | RC2   | RC1   | RC0   |
| PORTE | —                  | —                  | —     | —     | RE3 <sup>(2)</sup> | —     | —     | —     |

**Note 1:** Bits RB6 and RB7 read '1' while in Debug mode.

**2:** Bit PORTE3 is read-only, and will read '1' when MCLRE = 1 (Master Clear enabled).

**REGISTER 16-2: TRISx: TRI-STATE CONTROL REGISTER**

|         |         |         |         |         |         |         |         |
|---------|---------|---------|---------|---------|---------|---------|---------|
| R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
| TRISx7  | TRISx6  | TRISx5  | TRISx4  | TRISx3  | TRISx2  | TRISx1  | TRISx0  |
| bit 7   |         |         |         |         |         |         | bit 0   |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

bit 7-0 **TRISx<7:0>:** TRISx Port I/O Tri-state Control bits

1 = Port output driver is disabled

0 = Port output driver is enabled

**TABLE 16-3: TRIS REGISTERS**

| Name  | Bit 7                 | Bit 6                 | Bit 5  | Bit 4  | Bit 3  | Bit 2  | Bit 1  | Bit 0  |
|-------|-----------------------|-----------------------|--------|--------|--------|--------|--------|--------|
| TRISA | TRISA7                | TRISA6                | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 |
| TRISB | TRISB7 <sup>(1)</sup> | TRISB6 <sup>(1)</sup> | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 |
| TRISC | TRISC7                | TRISC6                | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 |

**Note 1:** Bits RB6 and RB7 read '1' while in Debug mode.

**REGISTER 16-3: LATx: LATx REGISTER<sup>(1)</sup>**

|         |         |         |         |         |         |         |         |
|---------|---------|---------|---------|---------|---------|---------|---------|
| R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
| LATx7   | LATx6   | LATx5   | LATx4   | LATx3   | LATx2   | LATx1   | LATx0   |
| bit 7   |         |         |         |         |         |         | bit 0   |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

bit 7-0      **LATx<7:0>:** Rx7:Rx0 Output Latch Value bits

**Note 1:** Writes to LATx are equivalent with writes to the corresponding PORTx register. Reads from LATx register return register values, not I/O pin values.

**TABLE 16-4: LAT REGISTERS**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| LATA | LATA7 | LATA6 | LATA5 | LATA4 | LATA3 | LATA2 | LATA1 | LATA0 |
| LATB | LATB7 | LATB6 | LATB5 | LATB4 | LATB3 | LATB2 | LATB1 | LATB0 |
| LATC | LATC7 | LATC6 | LATC5 | LATC4 | LATC3 | LATC2 | LATC1 | LATC0 |

**REGISTER 16-4: ANSELx: ANALOG SELECT REGISTER**

|         |         |         |         |         |         |         |         |
|---------|---------|---------|---------|---------|---------|---------|---------|
| R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
| ANSELx7 | ANSELx6 | ANSELx5 | ANSELx4 | ANSELx3 | ANSELx2 | ANSELx1 | ANSELx0 |
| bit 7   |         |         |         |         |         |         | bit 0   |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

bit 7-0 **ANSELx<7:0>**: Analog Select on Pins Rx<7:0>

1 = Digital Input buffers are disabled.

0 = ST and TTL input devices are enabled

**TABLE 16-5: ANALOG SELECT PORT REGISTERS**

| Name   | Bit 7   | Bit 6   | Bit 5   | Bit 4   | Bit 3   | Bit 2   | Bit 1   | Bit 0   |
|--------|---------|---------|---------|---------|---------|---------|---------|---------|
| ANSELA | ANSELA7 | ANSELA6 | ANSELA5 | ANSELA4 | ANSELA3 | ANSELA2 | ANSELA1 | ANSELA0 |
| ANSELB | ANSELB7 | ANSELB6 | ANSELB5 | ANSELB4 | ANSELB3 | ANSELB2 | ANSELB1 | ANSELB0 |
| ANSELC | ANSELC7 | ANSELC6 | ANSELC5 | ANSELC4 | ANSELC3 | ANSELC2 | ANSELC1 | ANSELC0 |



**REGISTER 16-5: WPUx: WEAK PULL-UP REGISTER**

|         |         |         |         |         |         |         |         |
|---------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| WPUx7   | WPUx6   | WPUx5   | WPUx4   | WPUx3   | WPUx2   | WPUx1   | WPUx0   |
| bit 7   |         |         |         |         |         |         | bit 0   |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

bit 7-0      **WPUx<7:0>:** Weak Pull-up PORTx Control bits

1 = Weak Pull-up enabled

0 = Weak Pull-up disabled

**TABLE 16-6: WEAK PULL-UP PORT REGISTERS**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3                | Bit 2 | Bit 1 | Bit 0 |
|------|-------|-------|-------|-------|----------------------|-------|-------|-------|
| WPUA | WPUA7 | WPUA6 | WPUA5 | WPUA4 | WPUA3                | WPUA2 | WPUA1 | WPUA0 |
| WPUB | WPUB7 | WPUB6 | WPUB5 | WPUB4 | WPUB3                | WPUB2 | WPUB1 | WPUB0 |
| WPUC | WPUC7 | WPUC6 | WPUC5 | WPUC4 | WPUC3                | WPUC2 | WPUC1 | WPUC0 |
| WPUE | —     | —     | —     | —     | WPUE3 <sup>(1)</sup> | —     | —     | —     |

**Note 1:** If MCLRE = 1, the weak pull-up in RE3 is always enabled; bit WPUE3 is not affected.

**REGISTER 16-6: ODCONx: OPEN-DRAIN CONTROL REGISTER**

|         |         |         |         |         |         |         |         |
|---------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| ODCx7   | ODCx6   | ODCx5   | ODCx4   | ODCx3   | ODCx2   | ODCx1   | ODCx0   |
| bit 7   |         |         |         |         |         |         | bit 0   |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

bit 7-0

**ODCx<7:0>:** Open-Drain Configuration on Pins Rx<7:0>

1 = Output drives only low-going signals (sink current only)

0 = Output drives both high-going and low-going signals (source and sink current)

**TABLE 16-7: OPEN-DRAIN CONTROL REGISTERS**

| Name   | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|--------|-------|-------|-------|-------|-------|-------|-------|-------|
| ODCONA | ODCA7 | ODCA6 | ODCA5 | ODCA4 | ODCA3 | ODCA2 | ODCA1 | ODCA0 |
| ODCONB | ODCB7 | ODCB6 | ODCB5 | ODCB4 | ODCB3 | ODCB2 | ODCB1 | ODCB0 |
| ODCONC | ODCC7 | ODCC6 | ODCC5 | ODCC4 | ODCC3 | ODCC2 | ODCC1 | ODCC0 |

**REGISTER 16-7: SLRCONx: SLEW RATE CONTROL REGISTER**

|         |         |         |         |         |         |         |         |
|---------|---------|---------|---------|---------|---------|---------|---------|
| R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
| SLRx7   | SLRx6   | SLRx5   | SLRx4   | SLRx3   | SLRx2   | SLRx1   | SLRx0   |
| bit 7   |         |         |         |         |         |         | bit 0   |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

bit 7-0

**SLRx<7:0>:** Slew Rate Control on Pins Rx<7:0>, respectively

1 = Port pin slew rate is limited

0 = Port pin slews at maximum rate

**TABLE 16-8: SLEW RATE CONTROL REGISTERS**

| Name    | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---------|-------|-------|-------|-------|-------|-------|-------|-------|
| SLRCONA | SLRA7 | SLRA6 | SLRA5 | SLRA4 | SLRA3 | SLRA2 | SLRA1 | SLRA0 |
| SLRCONB | SLRB7 | SLRB6 | SLRB5 | SLRB4 | SLRB3 | SLRB2 | SLRB1 | SLRB0 |
| SLRCONC | SLRC7 | SLRC6 | SLRC5 | SLRC4 | SLRC3 | SLRC2 | SLRC1 | SLRC0 |

## REGISTER 16-8: INLVLx: INPUT LEVEL CONTROL REGISTER

|         |         |         |         |         |         |         |         |
|---------|---------|---------|---------|---------|---------|---------|---------|
| R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
| INLVLA7 | INVLVB6 | INLVLC5 | INLVLE4 | INLVLE3 | INLVLE2 | INLVLE1 | INLVLE0 |
| bit 7   |         |         |         |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

bit 7-0

**INLVLx<7:0>**: Input Level Select on Pins Rx<7:0>, respectively

1 = ST input used for port reads and interrupt-on-change

0 = TTL input used for port reads and interrupt-on-change

## TABLE 16-9: INPUT LEVEL PORT REGISTERS

| Name   | Bit 7   | Bit 6   | Bit 5   | Bit 4                  | Bit 3                  | Bit 2                  | Bit 1                  | Bit 0   |
|--------|---------|---------|---------|------------------------|------------------------|------------------------|------------------------|---------|
| INLVLA | INLVLA7 | INLVLA6 | INLVLA5 | INLVLA4                | INLVLA3                | INLVLA2                | INLVLA1                | INLVLA0 |
| INVLVB | INVLVB7 | INVLVB6 | INVLVB5 | INVLVB4                | INVLVB3                | INVLVB2 <sup>(1)</sup> | INVLVB1 <sup>(1)</sup> | INVLVB0 |
| INLVLC | INLVLC7 | INLVLC6 | INLVLC5 | INLVLC4 <sup>(1)</sup> | INLVLC3 <sup>(1)</sup> | INLVLC2                | INLVLC1                | INLVLC0 |
| INLVLE | —       | —       | —       | —                      | INLVLE3                | —                      | —                      | —       |

**Note 1:** Any peripheral using the I<sup>2</sup>C pins read the I<sup>2</sup>C ST inputs when enabled via Rxyl2C.

**2:** Unimplemented in PIC18(L)F25K83.

**REGISTER 16-9: RxyI2C: I<sup>2</sup>C PAD Rxy CONTROL REGISTER**

|       |         |         |         |       |     |         |         |
|-------|---------|---------|---------|-------|-----|---------|---------|
| U-0   | R/W-0/0 | R/W-0/0 | R/W-0/0 | U-0   | U-0 | R/W-0/0 | R/W-0/0 |
| —     | SLEW    | PU<1:0> | —       | —     | —   | TH<1:0> | —       |
| bit 7 |         |         |         | bit 0 |     |         |         |

**Legend:**

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared | HS = Hardware set                                     |

|         |                                                                                                                                                                                                                                                                                 |
|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| bit 7   | <b>Unimplemented:</b> Read as '0'                                                                                                                                                                                                                                               |
| bit 6   | <b>SLEW:</b> I <sup>2</sup> C Specific Slew Rate Limiting is Enabled<br>1 = I <sup>2</sup> C specific slew rate limiting is enabled. Standard pad slew limiting is disabled. The SLRxy bit is ignored.<br>0 = Standard GPIO Slew Rate; enabled/disabled via SLRxy bit.          |
| bit 5-4 | <b>PU&lt;1:0&gt;:</b> I <sup>2</sup> C Pull-up Selection bits<br>11 = Reserved<br>10 = 10x current of standard weak pull-up<br>01 = 2x current of standard weak pull-up<br>00 = Standard GPIO weak pull-up, enabled via WPUxy bit                                               |
| bit 3-2 | <b>Unimplemented:</b> Read as '0'                                                                                                                                                                                                                                               |
| bit 1-0 | <b>TH&lt;1:0&gt;:</b> I <sup>2</sup> C Input Threshold Selection bits<br>11 = SMBus 3.0 (1.35 V) input threshold<br>10 = SMBus 2.0 (2.1 V) input threshold<br>01 = I <sup>2</sup> C specific input thresholds<br>00 = Standard GPIO Input pull-up, enabled via INLVLY registers |

**TABLE 16-10: I2C PAD CONTROL REGISTERS**

| Name   | Bit 7 | Bit 6 | Bit 5   | Bit 4 | Bit 3 | Bit 2 | Bit 1   | Bit 0 |
|--------|-------|-------|---------|-------|-------|-------|---------|-------|
| RB1I2C | —     | SLEW  | PU<1:0> | —     | —     | —     | TH<1:0> | —     |
| RB2I2C | —     | SLEW  | PU<1:0> | —     | —     | —     | TH<1:0> | —     |
| RC3I2C | —     | SLEW  | PU<1:0> | —     | —     | —     | TH<1:0> | —     |
| RC4I2C | —     | SLEW  | PU<1:0> | —     | —     | —     | TH<1:0> | —     |

**TABLE 16-11: SUMMARY OF REGISTERS ASSOCIATED WITH I/O**

| Name    | Bit 7                 | Bit 6                 | Bit 5   | Bit 4                  | Bit 3                  | Bit 2                  | Bit 1                  | Bit 0   | Register on Page    |
|---------|-----------------------|-----------------------|---------|------------------------|------------------------|------------------------|------------------------|---------|---------------------|
| PORTA   | RA7                   | RA6                   | RA5     | RA4                    | RA3                    | RA2                    | RA1                    | RA0     | <a href="#">253</a> |
| PORTB   | RB7 <sup>(1)</sup>    | RB6 <sup>(1)</sup>    | RB5     | RB4                    | RB3                    | RB2                    | RB1                    | RB0     | <a href="#">253</a> |
| PORTC   | RC7                   | RC6                   | RC5     | RC4                    | RC3                    | RC2                    | RC1                    | RC0     | <a href="#">253</a> |
| PORTE   | —                     | —                     | —       | —                      | RE3 <sup>(2)</sup>     | —                      | —                      | —       | <a href="#">253</a> |
| TRISA   | TRISA7                | TRISA6                | TRISA5  | TRISA4                 | TRISA3                 | TRISA2                 | TRISA1                 | TRISA0  | <a href="#">254</a> |
| TRISB   | TRISB7 <sup>(3)</sup> | TRISB6 <sup>(3)</sup> | TRISB5  | TRISB4                 | TRISB3                 | TRISB2                 | TRISB1                 | TRISB0  | <a href="#">254</a> |
| TRISC   | TRISC7                | TRISC6                | TRISC5  | TRISC4                 | TRISC3                 | TRISC2                 | TRISC1                 | TRISC0  | <a href="#">254</a> |
| LATA    | LATA7                 | LATA6                 | LATA5   | LATA4                  | LATA3                  | LATA2                  | LATA1                  | LATA0   | <a href="#">255</a> |
| LATB    | LATB7                 | LATB6                 | LATB5   | LATB4                  | LATB3                  | LATB2                  | LATB1                  | LATB0   | <a href="#">255</a> |
| LATC    | LATC7                 | LATC6                 | LATC5   | LATC4                  | LATC3                  | LATC2                  | LATC1                  | LATC0   | <a href="#">255</a> |
| ANSELA  | ANSELA7               | ANSELA6               | ANSELA5 | ANSELA4                | ANSELA3                | ANSELA2                | ANSELA1                | ANSELA0 | <a href="#">256</a> |
| ANSELB  | ANSELB7               | ANSELB6               | ANSELB5 | ANSELB4                | ANSELB3                | ANSELB2                | ANSELB1                | ANSELB0 | <a href="#">256</a> |
| ANSELC  | ANSELC7               | ANSELC6               | ANSELC5 | ANSELC4                | ANSELC3                | ANSELC2                | ANSELC1                | ANSELC0 | <a href="#">256</a> |
| WPUA    | WPUA7                 | WPUA6                 | WPUA5   | WPUA4                  | WPUA3                  | WPUA2                  | WPUA1                  | WPUA0   | <a href="#">257</a> |
| WPUB    | WPUB7                 | WPUB6                 | WPUB5   | WPUB4                  | WPUB3                  | WPUB2                  | WPUB1                  | WPUB0   | <a href="#">257</a> |
| WPUC    | WPUC7                 | WPUC6                 | WPUC5   | WPUC4                  | WPUC3                  | WPUC2                  | WPUC1                  | WPUC0   | <a href="#">257</a> |
| WPUE    | —                     | —                     | —       | —                      | WPUE3 <sup>(4)</sup>   | —                      | —                      | —       | <a href="#">257</a> |
| ODCONA  | ODCA7                 | ODCA6                 | ODCA5   | ODCA4                  | ODCA3                  | ODCA2                  | ODCA1                  | ODCA0   | <a href="#">258</a> |
| ODCONB  | ODCB7                 | ODCB6                 | ODCB5   | ODCB4                  | ODCB3                  | ODCB2                  | ODCB1                  | ODCB0   | <a href="#">258</a> |
| ODCONC  | ODCC7                 | ODCC6                 | ODCC5   | ODCC4                  | ODCC3                  | ODCC2                  | ODCC1                  | ODCC0   | <a href="#">258</a> |
| SLRCONA | SLRA7                 | SLRA6                 | SLRA5   | SLRA4                  | SLRA3                  | SLRA2                  | SLRA1                  | SLRA0   | <a href="#">259</a> |
| SLRCONB | SLRB7                 | SLRB6                 | SLRB5   | SLRB4                  | SLRB3                  | SLRB2                  | SLRB1                  | SLRB0   | <a href="#">259</a> |
| SLRCONC | SLRC7                 | SLRC6                 | SLRC5   | SLRC4                  | SLRC3                  | SLRC2                  | SLRC1                  | SLRC0   | <a href="#">259</a> |
| INLVLA  | INLVLA7               | INLVLA6               | INLVLA5 | INLVLA4                | INLVLA3                | INLVLA2                | INLVLA1                | INLVLA0 | <a href="#">260</a> |
| INVLVB  | INVLVB7               | INVLVB6               | INVLVB5 | INVLVB4                | INVLVB3                | INVLVB2 <sup>(5)</sup> | INVLVB1 <sup>(5)</sup> | INVLVB0 | <a href="#">260</a> |
| INLVLC  | INLVLC7               | INLVLC6               | INLVLC5 | INLVLC4 <sup>(5)</sup> | INLVLC3 <sup>(5)</sup> | INLVLC2                | INLVLC1                | INLVLC0 | <a href="#">260</a> |
| INLVLE  | —                     | —                     | —       | —                      | INLVLE3                | —                      | —                      | —       | <a href="#">260</a> |
| RB1I2C  | —                     | SLEW                  | PU<1:0> |                        | —                      | —                      | TH<1:0>                |         | <a href="#">261</a> |
| RB2I2C  | —                     | SLEW                  | PU<1:0> |                        | —                      | —                      | TH<1:0>                |         | <a href="#">261</a> |
| RC3I2C  | —                     | SLEW                  | PU<1:0> |                        | —                      | —                      | TH<1:0>                |         | <a href="#">261</a> |
| RC4I2C  | —                     | SLEW                  | PU<1:0> |                        | —                      | —                      | TH<1:0>                |         | <a href="#">261</a> |

**Legend:** — = unimplemented location, read as '0'. Shaded cells are not used by I/O Ports.

- Note**
- 1: Bits RB6 and RB7 read '1' while in Debug mode.
  - 2: Bit PORTE3 is read-only, and will read '1' when MCLRE = 1 (Master Clear enabled).
  - 3: Bits RB6 and RB7 read '1' while in Debug mode.
  - 4: If MCLRE = 1, the weak pull-up in RE3 is always enabled; bit WPUE3 is not affected.
  - 5: Any peripheral using the I<sup>2</sup>C pins read the I<sup>2</sup>C ST inputs when enabled via RxyI2C.

## 17.0 PERIPHERAL PIN SELECT (PPS) MODULE

The Peripheral Pin Select (PPS) module connects peripheral inputs and outputs to the device I/O pins. Only digital signals are included in the selections. All analog inputs and outputs remain fixed to their assigned pins. Input and output selections are independent as shown in the simplified block diagram [Figure 17-1](#).

The peripheral input is selected with the peripheral xxxPPS register ([Register 17-1](#)), and the peripheral output is selected with the PORT RxyPPS register ([Register 17-2](#)). For example, to select PORTC<7> as the UART1 RX input, set U1RXPPS to 0b1 0111, and to select PORTC<6> as the UART1 TX output set RC6PPS to 0b01 0011.

## 17.1 PPS Inputs

Each peripheral has a PPS register with which the inputs to the peripheral are selected. Inputs include the device pins.

Multiple peripherals can operate from the same source simultaneously. Port reads always return the pin level regardless of peripheral PPS selection. If a pin also has analog functions associated, the ANSEL bit for that pin must be cleared to enable the digital input buffer.

Although every peripheral has its own PPS input selection register, the selections are identical for every peripheral as shown in [Register 17-1](#).

**Note:** The notation “xxx” in the register name is a place holder for the peripheral identifier. For example, INT0PPS.

## 17.2 PPS Outputs

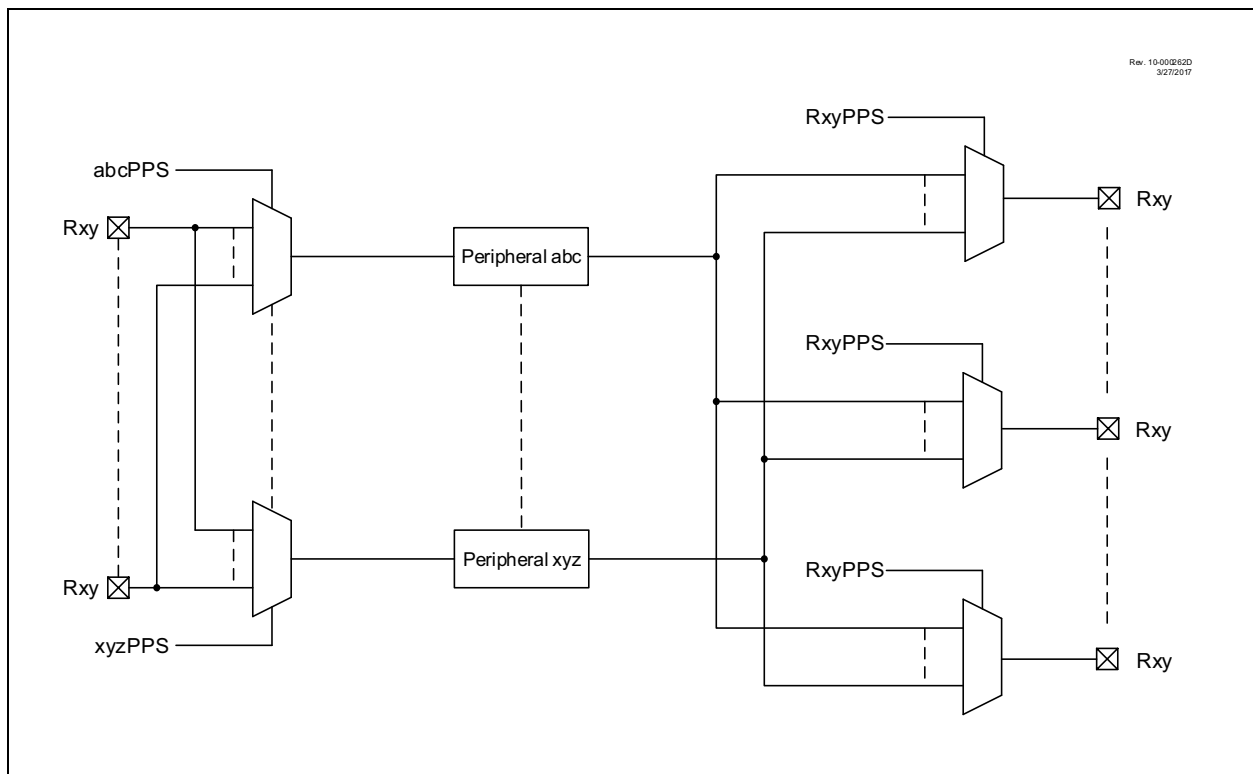
Each I/O pin has a PPS register with which the pin output source is selected. With few exceptions, the port TRIS control associated with that pin retains control over the pin output driver. Peripherals that control the pin output driver as part of the peripheral operation will override the TRIS control as needed. These peripherals include:

- UART

Although every pin has its own PPS peripheral selection register, the selections are identical for every pin as shown in [Register 17-2](#).

**Note:** The notation “Rxy” is a place holder for the pin identifier. For example, RA0PPS.

### FIGURE 17-1: SIMPLIFIED PPS BLOCK DIAGRAM



## 17.3 Bidirectional Pins

PPS selections for peripherals with bidirectional signals on a single pin must be made so that the PPS input and PPS output select the same pin. Peripherals that have bidirectional signals include:

- I<sup>2</sup>C

**Note:** Refer to [Table 17-1](#) for pins that are I<sup>2</sup>C compatible. Clock and data signals can be routed to any pin, however pins without I<sup>2</sup>C compatibility will operate at standard TTL/ST logic levels as selected by the INVLV register.

## 17.4 PPS Lock

The PPS includes a mode in which all input and output selections can be locked to prevent inadvertent changes. PPS selections are locked by setting the PPSLOCKED bit of the PPSLOCK register. Setting and clearing this bit requires a special sequence as an extra precaution against inadvertent changes. Examples of setting and clearing the PPSLOCKED bit are shown in [Example 17-1](#).

### EXAMPLE 17-1: PPS LOCK SEQUENCE

```
; Disable interrupts:
BCF      INTCON0,GIE

; Bank to PPSLOCK register
BANKSEL  PPSLOCK
MOVLB    PPSLOCK
MOVLW    55h

; Required sequence, next 4 instructions
MOVWF    PPSLOCK
MOVLW    AAh
MOVWF    PPSLOCK

; Set PPSLOCKED bit to disable writes
; Only a BSF instruction will work
BSF      PPSLOCK,0

; Enable Interrupts
BSF      INTCON0,GIE
```

### EXAMPLE 17-2: PPS UNLOCK SEQUENCE

```
; Disable interrupts:
BCF      INTCON0,GIE

; Bank to PPSLOCK register
BANKSEL  PPSLOCK
MOVLB    PPSLOCK
MOVLW    55h

; Required sequence, next 4 instructions
MOVWF    PPSLOCK
MOVLW    AAh
MOVWF    PPSLOCK

; Clear PPSLOCKED bit to enable writes
; Only a BCF instruction will work
BCF      PPSLOCK,0

; Enable Interrupts
BSF      INTCON0,GIE
```

## 17.5 PPS One-way Lock

When this bit is set, the PPSLOCKED bit can only be cleared and set one time after a device Reset. This allows for clearing the PPSLOCKED bit so that the input and output selections can be made during initialization. When the PPSLOCKED bit is set after all selections have been made, it will remain set and cannot be cleared until after the next device Reset event.

## 17.6 Operation During Sleep

PPS input and output selections are unaffected by Sleep.

## 17.7 Effects of a Reset

A device Power-on-Reset (POR) clears all PPS input and output selections to their default values. All other Resets leave the selections unchanged. Default input selections are shown in pin allocation [Table 3](#). The PPS one-way lock is also removed.



## 17.8 Register Definitions: PPS Input Selection

**REGISTER 17-1: xxxPPS: PERIPHERAL xxx INPUT SELECTION**

| U-0   | U-0 | R/W-m/u <sup>(1)</sup> | R/W-m/u <sup>(1)</sup> | R/W-m/u <sup>(1)</sup> | R/W-m/u <sup>(1)</sup> | R/W-m/u <sup>(1)</sup> | R/W-m/u <sup>(1)</sup> |
|-------|-----|------------------------|------------------------|------------------------|------------------------|------------------------|------------------------|
| —     | —   | xxxPPS<5:0>            |                        |                        |                        |                        |                        |
| bit 7 |     |                        |                        |                        |                        |                        |                        |
|       |     |                        |                        |                        |                        |                        | bit 0                  |

### Legend:

|                      |                                    |                                                       |
|----------------------|------------------------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit                   | -n/n = Value at POR and BOR/Value at all other Resets |
| u = Bit is unchanged | x = Bit is unknown                 | q = value depends on peripheral                       |
| '1' = Bit is set     | U = Unimplemented bit, read as '0' | m = value depends on default location for that input  |
| '0' = Bit is cleared |                                    |                                                       |

bit 7-6 **Unimplemented:** Read as '0'

bit 5-3 **xxxPPS<5:3>:** Peripheral xxx Input PORTx Pin Selection bits  
 See [Table 17-1](#) for the list of available ports and default pin locations.  
 101 = Reserved  
 100 = Reserved  
 011 = Reserved  
 010 = PORTC  
 001 = PORTB  
 000 = PORTA

bit 2-0 **xxxPPS<2:0>:** Peripheral xxx Input PORTx Pin Selection bits  
 111 = Peripheral input is from PORTx Pin 7 (Rx7)  
 110 = Peripheral input is from PORTx Pin 6 (Rx6)  
 101 = Peripheral input is from PORTx Pin 5 (Rx5)  
 100 = Peripheral input is from PORTx Pin 4 (Rx4)  
 011 = Peripheral input is from PORTx Pin 3 (Rx3)  
 010 = Peripheral input is from PORTx Pin 2 (Rx2)  
 001 = Peripheral input is from PORTx Pin 1 (Rx1)  
 000 = Peripheral input is from PORTx Pin 0 (Rx0)

**Note 1:** The Reset value 'm' of this register is determined by device default locations for that input.

**TABLE 17-1: PPS INPUT REGISTER DETAILS**

| Peripheral              | PPS Input Register | Default Pin Selection at POR | Register Reset Value at POR | Input Available from Selected PORTx |   |   |
|-------------------------|--------------------|------------------------------|-----------------------------|-------------------------------------|---|---|
|                         |                    |                              |                             | PIC18(L)F2xK83                      |   |   |
| Interrupt 0             | INT0PPS            | RB0                          | 0b0 1000                    | A                                   | B | — |
| Interrupt 1             | INT1PPS            | RB1                          | 0b0 1001                    | A                                   | B | — |
| Interrupt 2             | INT2PPS            | RB2                          | 0b0 1010                    | A                                   | B | — |
| Timer0 Clock            | T0CKIPPS           | RA4                          | 0b0 0100                    | A                                   | B | — |
| Timer1 Clock            | T1CKIPPS           | RC0                          | 0b1 0000                    | A                                   | — | C |
| Timer1 Gate             | T1GPPS             | RB5                          | 0b0 1101                    | —                                   | B | C |
| Timer3 Clock            | T3CKIPPS           | RC0                          | 0b1 0000                    | —                                   | B | C |
| Timer3 Gate             | T3GPPS             | RC0                          | 0b1 0000                    | A                                   | — | C |
| Timer5 Clock            | T5CKIPPS           | RC2                          | 0b1 0010                    | A                                   | — | C |
| Timer5 Gate             | T5GPPS             | RB4                          | 0b0 1100                    | —                                   | B | C |
| Timer2 Clock            | T2INPPS            | RC3                          | 0b1 0011                    | A                                   | — | C |
| Timer4 Clock            | T4INPPS            | RC5                          | 0b1 0101                    | —                                   | B | C |
| Timer6 Clock            | T6INPPS            | RB7                          | 0b0 1111                    | —                                   | B | C |
| CCP1                    | CCP1PPS            | RC2                          | 0b1 0010                    | —                                   | B | C |
| CCP2                    | CCP2PPS            | RC1                          | 0b1 0001                    | —                                   | B | C |
| CCP3                    | CCP3PPS            | RB5                          | 0b0 1101                    | —                                   | B | C |
| CCP4                    | CCP4PPS            | RB0                          | 0b0 1000                    | —                                   | B | C |
| SMT1 Window             | SMT1WINPPS         | RC0                          | 0b1 0000                    | —                                   | B | C |
| SMT1 Signal             | SMT1SIGPPS         | RB4                          | 0b0 1100                    | —                                   | B | C |
| SMT2 Window             | SMT2WINPPS         | RB5                          | 0b0 1101                    | —                                   | B | C |
| SMT2 Signal             | SMT2SIGPPS         | RC1                          | 0b1 0001                    | —                                   | B | C |
| CWG1                    | CWG1PPS            | RB0                          | 0b0 1000                    | —                                   | B | C |
| CWG2                    | CWG2PPS            | RB1                          | 0b0 1001                    | —                                   | B | C |
| CWG3                    | CWG3PPS            | RB2                          | 0b0 1010                    | —                                   | B | C |
| DSM1 Carrier Low        | MD1CARLPPS         | RA3                          | 0b0 0011                    | A                                   | — | C |
| DSM1 Carrier High       | MD1CARHPPS         | RA4                          | 0b0 0100                    | A                                   | — | C |
| DSM1 Source             | MD1SRCPPS          | RA5                          | 0b0 0101                    | A                                   | — | C |
| CLCx Input 1            | CLCIN0PPS          | RA0                          | 0b0 0000                    | A                                   | — | C |
| CLCx Input 2            | CLCIN1PPS          | RA1                          | 0b0 0001                    | A                                   | — | C |
| CLCx Input 3            | CLCIN2PPS          | RB6                          | 0b0 1110                    | —                                   | B | C |
| CLCx Input 4            | CLCIN3PPS          | RB7                          | 0b0 1111                    | —                                   | B | C |
| ADC Conversion Trigger  | ADACTPPS           | RB4                          | 0b0 1100                    | —                                   | B | C |
| SPI1 Clock              | SPI1SCKPPS         | RC3                          | 0b1 0011                    | —                                   | B | C |
| SPI1 Data               | SPI1SDIPPS         | RC4                          | 0b1 0100                    | —                                   | B | C |
| SPI1 Slave Select       | SPI1SSPPS          | RA5                          | 0b0 0101                    | A                                   | — | C |
| I <sup>2</sup> C1 Clock | I2C1SCLPPS         | RC3                          | 0b1 0011                    | —                                   | B | C |
| I <sup>2</sup> C1 Data  | I2C1SDAPPS         | RC4                          | 0b1 0100                    | —                                   | B | C |
| I <sup>2</sup> C2 Clock | I2C2SCLPPS         | RB1                          | 0b0 1001                    | —                                   | B | C |
| I <sup>2</sup> C2 Data  | I2C2SDAPPS         | RB2                          | 0b0 1010                    | —                                   | B | C |
| UART1 Receive           | U1RXPPS            | RC7                          | 0b1 0111                    | —                                   | B | C |
| UART1 Clear To Send     | U1CTSPPS           | RC6                          | 0b1 0110                    | —                                   | B | C |
| UART2 Receive           | U2RXPPS            | RB7                          | 0b0 1111                    | —                                   | B | C |
| UART2 Clear To Send     | U2CTSPPS           | RB6                          | 0b0 1110                    | —                                   | B | C |
| CAN Receive             | CANRXPPS           | RB3                          | 0b0 1011                    | —                                   | B | C |

**REGISTER 17-2: RxyPPS: PIN Rxy OUTPUT SOURCE SELECTION REGISTER**

| U-0   | U-0 | R/W-0/u     | R/W-0/u | R/W-0/u | R/W-0/u | R/W-0/u | R/W-0/u |
|-------|-----|-------------|---------|---------|---------|---------|---------|
| —     | —   | RxyPPS<5:0> |         |         |         |         |         |
| bit 7 |     |             |         |         |         |         | bit 0   |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-6

**Unimplemented:** Read as '0'

bit 5-0

**RxyPPS<5:0>:** Pin Rxy Output Source Selection bits

See [Table 17-2](#) for the list of available ports.

**TABLE 17-2: PPS OUTPUT REGISTER DETAILS**

| RxyPPS<5:0>              | Pin Rxy<br>Output<br>Source | Device Configuration |   |   |
|--------------------------|-----------------------------|----------------------|---|---|
|                          |                             | PIC18(L)F2xK83       |   |   |
| 0b11 1111 -<br>0b11 0101 | Reserved                    |                      |   |   |
| 0b11 0100                | CANTX1                      | —                    | B | C |
| 0b11 0011                | CANTX0                      | —                    | B | C |
| 0b11 0010                | ADGRDB                      | A                    | — | C |
| 0b11 0001                | ADGRDA                      | A                    | — | C |
| 0b11 0000                | CWG3D                       | A                    | — | C |
| 0b10 1111                | CWG3C                       | A                    | — | C |
| 0b10 1110                | CWG3B                       | A                    | — | C |
| 0b10 1101                | CWG3A                       | —                    | B | C |
| 0b10 1100                | CWG2D                       | —                    | B | C |
| 0b10 1011                | CWG2C                       | —                    | B | C |
| 0b10 1010                | CWG2B                       | —                    | B | C |
| 0b10 1001                | CWG2A                       | —                    | B | C |
| 0b10 1000                | DSM1                        | A                    | — | C |
| 0b10 0111                | CLKR                        | —                    | B | C |
| 0b10 0110                | NCO1                        | A                    | — | C |
| 0b10 0101                | TMR0                        | —                    | B | C |
| 0b10 0100                | I <sup>2</sup> C2 (SDA)     | —                    | B | C |
| 0b10 0011                | I <sup>2</sup> C2 (SCL)     | —                    | B | C |
| 0b10 0010                | I <sup>2</sup> C1 (SDA)     | —                    | B | C |
| 0b10 0001                | I <sup>2</sup> C1 (SCL)     | —                    | B | C |
| 0b10 0000                | SPI1 (SS)                   | A                    | — | C |
| 0b01 1111                | SPI1 (SDO)                  | —                    | B | C |
| 0b01 1110                | SPI1 (SCK)                  | —                    | B | C |
| 0b01 1101                | C2OUT                       | A                    | — | C |
| 0b01 1100                | C1OUT                       | A                    | — | C |
| 0b01 1011 -<br>0b01 1001 | Reserved                    |                      |   |   |
| 0b01 1000                | UART2 (RTS)                 | —                    | B | C |
| 0b01 0111                | UART2 (TXDE)                | —                    | B | C |
| 0b01 0110                | UART2 (TX)                  | —                    | B | C |
| 0b01 0101                | UART1 (RTS)                 | —                    | B | C |
| 0b01 0100                | UART1 (TXDE)                | —                    | B | C |
| 0b01 0011                | UART1 (TX)                  | —                    | B | C |
| 0b01 0010 -<br>0b01 0001 | Reserved                    |                      |   |   |
| 0b01 0000                | PWM8                        | A                    | — | C |
| 0b00 1111                | PWM7                        | A                    | — | C |
| 0b00 1110                | PWM6                        | A                    | — | C |
| 0b00 1101                | PWM5                        | A                    | — | C |
| 0b00 1100                | CCP4                        | —                    | B | C |
| 0b00 1011                | CCP3                        | —                    | B | C |
| 0b00 1010                | CCP2                        | —                    | B | C |
| 0b00 1001                | CCP1                        | —                    | B | C |
| 0b00 1000                | CWG1D                       | —                    | B | C |
| 0b00 0111                | CWG1C                       | —                    | B | C |
| 0b00 0110                | CWG1B                       | —                    | B | C |

**TABLE 17-2: PPS OUTPUT REGISTER DETAILS**

| RxyPPS<5:0> | Pin Rxy Output Source | Device Configuration |   |   |
|-------------|-----------------------|----------------------|---|---|
|             |                       | PIC18(L)F2xK83       |   |   |
| 0b00 0101   | CWG1A                 | —                    | B | C |
| 0b00 0100   | CLC4OUT               | —                    | B | C |
| 0b00 0011   | CLC3OUT               | —                    | B | C |
| 0b00 0010   | CLC2OUT               | A                    | — | C |
| 0b00 0001   | CLC1OUT               | A                    | — | C |
| 0b00 0000   | LATxy                 | A                    | B | C |

## REGISTER 17-3: PPSLOCK: PPS LOCK REGISTER

|       |     |     |     |     |     |     |           |
|-------|-----|-----|-----|-----|-----|-----|-----------|
| U-0   | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0/0   |
| —     | —   | —   | —   | —   | —   | —   | PPSLOCKED |
| bit 7 |     |     |     |     |     |     | bit 0     |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-1

**Unimplemented:** Read as '0'

bit 0

**PPSLOCKED:** PPS Locked bit

1 = PPS is locked.

0 = PPS is not locked. PPS selections can be changed.

**TABLE 17-3: SUMMARY OF REGISTERS ASSOCIATED WITH THE PPS MODULE**

| Name       | Bit 7 | Bit 6 | Bit 5 | Bit 4           | Bit 3 | Bit 2 | Bit 1 | Bit 0     | Register on page |
|------------|-------|-------|-------|-----------------|-------|-------|-------|-----------|------------------|
| PPSLOCK    | —     | —     | —     | —               | —     | —     | —     | PPSLOCKED | 269              |
| INT0PPS    | —     | —     | —     | INT0PPS<4:0>    |       |       |       |           | 265              |
| INT1PPS    | —     | —     | —     | INT1PPS<4:0>    |       |       |       |           | 265              |
| INT2PPS    | —     | —     | —     | INT2PPS<4:0>    |       |       |       |           | 265              |
| T0CKIPPS   | —     | —     | —     | T0CKIPPS<4:0>   |       |       |       |           | 265              |
| T1CKIPPS   | —     | —     | —     | T1CKIPPS<4:0>   |       |       |       |           | 265              |
| T1GPPS     | —     | —     | —     | T1GPPS<4:0>     |       |       |       |           | 265              |
| T3CKIPPS   | —     | —     | —     | T3CKIPPS<4:0>   |       |       |       |           | 265              |
| T3GPPS     | —     | —     | —     | T3GPPS<4:0>     |       |       |       |           | 265              |
| T5CKIPPS   | —     | —     | —     | T5CKIPPS<4:0>   |       |       |       |           | 265              |
| T5GPPS     | —     | —     | —     | T5GPPS<4:0>     |       |       |       |           | 265              |
| T2INPPS    | —     | —     | —     | T2INPPS<4:0>    |       |       |       |           | 265              |
| T4INPPS    | —     | —     | —     | T4INPPS<4:0>    |       |       |       |           | 265              |
| T6INPPS    | —     | —     | —     | T6INPPS<4:0>    |       |       |       |           | 265              |
| CCP1PPS    | —     | —     | —     | CCP1PPS<4:0>    |       |       |       |           | 265              |
| CCP2PPS    | —     | —     | —     | CCP2PPS<4:0>    |       |       |       |           | 265              |
| CCP3PPS    | —     | —     | —     | CCP3PPS<4:0>    |       |       |       |           | 265              |
| CCP4PPS    | —     | —     | —     | CCP4PPS<4:0>    |       |       |       |           | 265              |
| SMT1WINPPS | —     | —     | —     | SMT1WINPPS<4:0> |       |       |       |           | 265              |
| SMT1SIGPPS | —     | —     | —     | SMT1SIGPPS<4:0> |       |       |       |           | 265              |
| SMT2WINPPS | —     | —     | —     | SMT2WINPPS<4:0> |       |       |       |           | 265              |
| SMT2SIGPPS | —     | —     | —     | SMT2SIGPPS<4:0> |       |       |       |           | 265              |
| CWG1PPS    | —     | —     | —     | CWG1PPS<4:0>    |       |       |       |           | 265              |
| CWG2PPS    | —     | —     | —     | CWG2PPS<4:0>    |       |       |       |           | 265              |
| CWG3PPS    | —     | —     | —     | CWG3PPS<4:0>    |       |       |       |           | 265              |
| MD1CARLPPS | —     | —     | —     | MDCARLPPS<4:0>  |       |       |       |           | 265              |
| MD1CARHPPS | —     | —     | —     | MDCARHPPS<4:0>  |       |       |       |           | 265              |
| MD1SRCPPS  | —     | —     | —     | MDSRCPPS<4:0>   |       |       |       |           | 265              |
| CLCIN0PPS  | —     | —     | —     | CLCIN0PPS<4:0>  |       |       |       |           | 265              |
| CLCIN1PPS  | —     | —     | —     | CLCIN1PPS<4:0>  |       |       |       |           | 265              |
| CLCIN2PPS  | —     | —     | —     | CLCIN2PPS<4:0>  |       |       |       |           | 265              |
| CLCIN3PPS  | —     | —     | —     | CLCIN3PPS<4:0>  |       |       |       |           | 265              |
| ADACTPPS   | —     | —     | —     | ADACTPPS<4:0>   |       |       |       |           | 265              |
| SPI1SCKPPS | —     | —     | —     | SPI1SCKPPS<4:0> |       |       |       |           | 265              |
| SPI1SDIPPS | —     | —     | —     | SPI1SDIPPS<4:0> |       |       |       |           | 265              |
| SPI1SSPPS  | —     | —     | —     | SPI1SSPPS<4:0>  |       |       |       |           | 265              |
| I2C1SCLPPS | —     | —     | —     | I2C1SCLPPS<4:0> |       |       |       |           | 265              |
| I2C1SDAPPS | —     | —     | —     | I2C1SDAPPS<4:0> |       |       |       |           | 265              |
| I2C2SCLPPS | —     | —     | —     | I2C2SCLPPS<4:0> |       |       |       |           | 265              |
| I2C2SDAPPS | —     | —     | —     | I2C2SDAPPS<4:0> |       |       |       |           | 265              |
| U1RXPPS    | —     | —     | —     | U1RXPPS<4:0>    |       |       |       |           | 265              |
| U1CTSPPS   | —     | —     | —     | U1CTSPPS<4:0>   |       |       |       |           | 265              |
| U2RXPPS    | —     | —     | —     | U2RXPPS<4:0>    |       |       |       |           | 265              |
| U2CTSPPS   | —     | —     | —     | U2CTPPS<4:0>    |       |       |       |           | 265              |
| RxyPPS     | —     | —     | —     | RxyPPS<4:0>     |       |       |       |           | 265              |
| CANRXPPS   | —     | —     | —     | CANRXPPS<4:0>   |       |       |       |           | 265              |

**Legend:** — = unimplemented, read as '0'. Shaded cells are unused by the PPS module.

## 18.0 INTERRUPT-ON-CHANGE

PORTA, PORTB, PORTC and pin RE3 of PORTE can be configured to operate as Interrupt-on-Change (IOC) pins on PIC18(L)F25/26K83 family devices. An interrupt can be generated by detecting a signal that has either a rising edge or a falling edge. Any individual port pin, or combination of port pins, can be configured to generate an interrupt. The interrupt-on-change module has the following features:

- Interrupt-on-Change enable (Master Switch)
- Individual pin configuration
- Rising and falling edge detection
- Individual pin interrupt flags

Figure 18-1 is a block diagram of the IOC module.

### 18.1 Enabling the Module

To allow individual port pins to generate an interrupt, the IOCIE bit of the PIE0 register must be set. If the IOCIE bit is disabled, the edge detection on the pin will still occur, but an interrupt will not be generated.

### 18.2 Individual Pin Configuration

For each port pin, a rising edge detector and a falling edge detector are present. To enable a pin to detect a rising edge, the associated bit of the IOCxP register is set. To enable a pin to detect a falling edge, the associated bit of the IOCxN register is set.

A pin can be configured to detect rising and falling edges simultaneously by setting both associated bits of the IOCxP and IOCxN registers, respectively.

## 18.3 Interrupt Flags

The IOCAFx, IOCBFx, IOCCFx and IOCEF3 bits located in the IOCAF, IOCBF, IOCCF and IOCEF registers respectively, are status flags that correspond to the interrupt-on-change pins of the associated port. If an expected edge is detected on an appropriately enabled pin, then the status flag for that pin will be set, and an interrupt will be generated if the IOCIE bit is set. The IOCIF bit of the PIR0 register reflects the status of all IOCAFx, IOCBFx, IOCCFx and IOCEF3 bits.

### 18.4 Clearing Interrupt Flags

The individual status flags, (IOCAFx, IOCBFx, IOCCFx and IOCEF3 bits), can be cleared by resetting them to zero. If another edge is detected during this clearing operation, the associated status flag will be set at the end of the sequence, regardless of the value actually being written.

In order to ensure that no detected edge is lost while clearing flags, only AND operations masking out known changed bits should be performed. The following sequence is an example of what should be performed.

#### EXAMPLE 18-1: CLEARING INTERRUPT FLAGS (PORTA EXAMPLE)

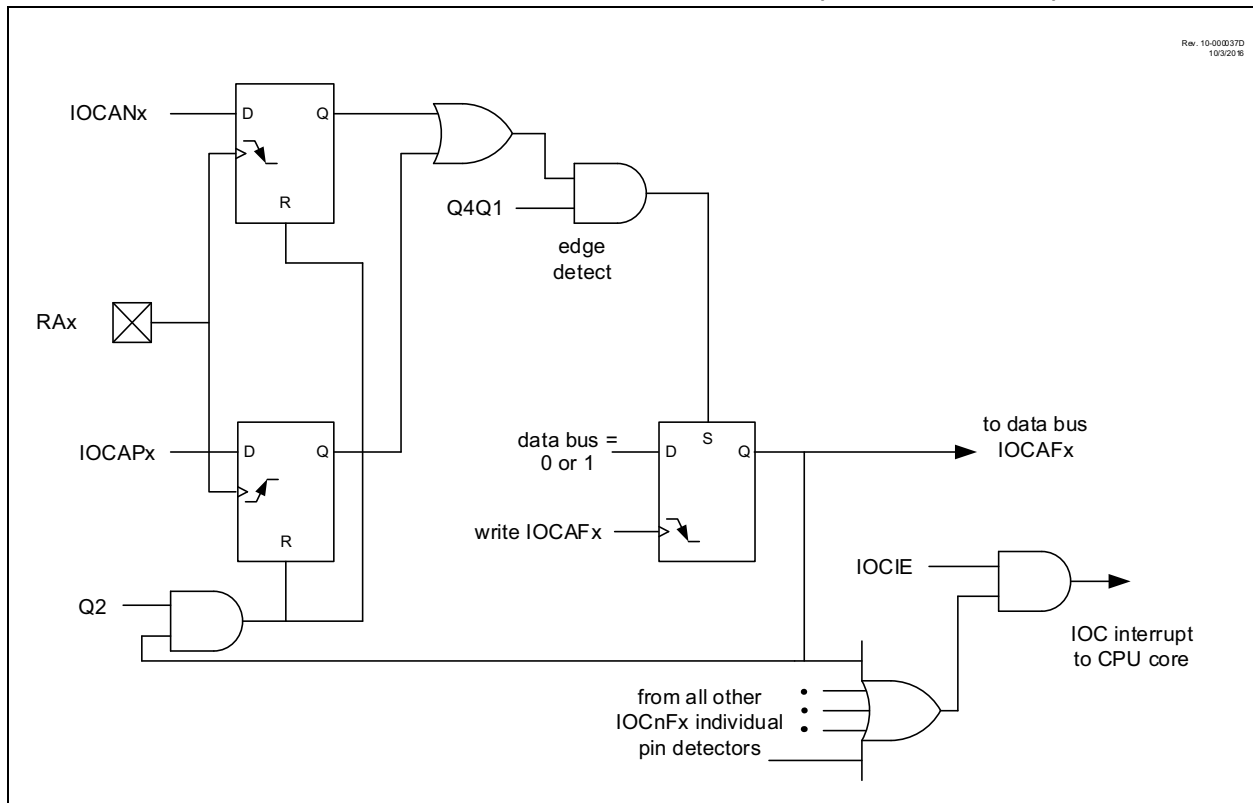
```
MOVLW    0xff
XORWF    IOCAF, W
ANDWF    IOCAF, F
```

### 18.5 Operation in Sleep

The interrupt-on-change interrupt sequence will wake the device from Sleep mode, if the IOCIE bit is set.

If an edge is detected while in Sleep mode, the IOCxF register will be updated prior to the first instruction executed out of Sleep.

**FIGURE 18-1: INTERRUPT-ON-CHANGE BLOCK DIAGRAM (PORTA EXAMPLE)**





## 18.6 Register Definitions: Interrupt-on-Change Control

### REGISTER 18-1: IOCxP: INTERRUPT-ON-CHANGE POSITIVE EDGE REGISTER EXAMPLE

|         |         |         |         |         |         |         |         |
|---------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| IOCxP7  | IOCxP6  | IOCxP5  | IOCxP4  | IOCxP3  | IOCxP2  | IOCxP1  | IOCxP0  |
| bit 7   |         |         |         |         |         |         | bit 0   |

#### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-0

**IOCxP<7:0>:** Interrupt-on-Change Positive Edge Enable bits

1 = Interrupt-on-Change enabled on the IOCx pin for a positive-going edge. Associated Status bit and interrupt flag will be set upon detecting an edge.

0 = Interrupt-on-Change disabled for the associated pin.

### REGISTER 18-2: IOCxN: INTERRUPT-ON-CHANGE NEGATIVE EDGE REGISTER EXAMPLE

|         |         |         |         |         |         |         |         |
|---------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| IOCxN7  | IOCxN6  | IOCxN5  | IOCxN4  | IOCxN3  | IOCxN2  | IOCxN1  | IOCxN0  |
| bit 7   |         |         |         |         |         |         | bit 0   |

#### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-0

**IOCxN<7:0>:** Interrupt-on-Change Negative Edge Enable bits

1 = Interrupt-on-Change enabled on the IOCx pin for a negative-going edge. Associated Status bit and interrupt flag will be set upon detecting an edge.

0 = Interrupt-on-Change disabled for the associated pin

### REGISTER 18-3: IOCxF: INTERRUPT-ON-CHANGE FLAG REGISTER EXAMPLE

|            |            |            |            |            |            |            |            |
|------------|------------|------------|------------|------------|------------|------------|------------|
| R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 |
| IOCxF7     | IOCxF6     | IOCxF5     | IOCxF4     | IOCxF3     | IOCxF2     | IOCxF1     | IOCxF0     |
| bit 7      |            |            |            |            |            |            | bit 0      |

#### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

HS - Bit is set in hardware

bit 7-0

**IOCxF<7:0>:** Interrupt-on-Change Flag bits

1 = A enabled change was detected on the associated pin. Set when IOCxP[n] = 1 and a positive edge was detected on the IOCn pin, or when IOCxN[n] = 1 and a negative edge was detected on the IOCn pin

0 = No change was detected, or the user cleared the detected change

**TABLE 18-1: IOC REGISTERS**

| Name  | Bit 7  | Bit 6  | Bit 5  | Bit 4  | Bit 3                 | Bit 2  | Bit 1  | Bit 0  |
|-------|--------|--------|--------|--------|-----------------------|--------|--------|--------|
| IOCAP | IOCAP7 | IOCAP6 | IOCAP5 | IOCAP4 | IOCAP3                | IOCAP2 | IOCAP1 | IOCAP0 |
| IOCAN | IOCAN7 | IOCAN6 | IOCAN5 | IOCAN4 | IOCAN3                | IOCAN2 | IOCAN1 | IOCAN0 |
| IOCAF | IOCAF7 | IOCAF6 | IOCAF5 | IOCAF4 | IOCAF3                | IOCAF2 | IOCAF1 | IOCAF0 |
| IOCBP | IOCBP7 | IOCBP6 | IOCBP5 | IOCBP4 | IOCBP3                | IOCBP2 | IOCBP1 | IOCBP0 |
| IOCBN | IOCBN7 | IOCBN6 | IOCBN5 | IOCBN4 | IOCBN3                | IOCBN2 | IOCBN1 | IOCBN0 |
| IOCBF | IOCBF7 | IOCBF6 | IOCBF5 | IOCBF4 | IOCBF3                | IOCBF2 | IOCBF1 | IOCBF0 |
| IOCCP | IOCCP7 | IOCCP6 | IOCCP5 | IOCCP4 | IOCCP3                | IOCCP2 | IOCCP1 | IOCCP0 |
| IOCCN | IOCCN7 | IOCCN6 | IOCCN5 | IOCCN4 | IOCCN3                | IOCCN2 | IOCCN1 | IOCCN0 |
| IOCCF | IOCCF7 | IOCCF6 | IOCCF5 | IOCCF4 | IOCCF3                | IOCCF2 | IOCCF1 | IOCCF0 |
| IOCEP | —      | —      | —      | —      | IOCEP3 <sup>(1)</sup> | —      | —      | —      |
| IOCEN | —      | —      | —      | —      | IOCEN3 <sup>(1)</sup> | —      | —      | —      |
| IOCEF | —      | —      | —      | —      | IOCEF3 <sup>(1)</sup> | —      | —      | —      |

**Note 1:** If MCLRE = 1 or LVP = 1, RE3 port functionality is disabled and IOC on RE3 is not available.

**TABLE 18-2: SUMMARY OF REGISTERS ASSOCIATED WITH INTERRUPT-ON-CHANGE**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page    |
|------|-------|-------|-------|-------|-------|-------|-------|-------|---------------------|
| IOCF | IOCF7 | IOCF6 | IOCF5 | IOCF4 | IOCF3 | IOCF2 | IOCF1 | IOCF0 | <a href="#">273</a> |
| IOCN | IOCN7 | IOCN6 | IOCN5 | IOCN4 | IOCN3 | IOCN2 | IOCN1 | IOCN0 | <a href="#">273</a> |
| IOCP | IOCP7 | IOCP6 | IOCP5 | IOCP4 | IOCP3 | IOCP2 | IOCP1 | IOCP0 | <a href="#">273</a> |

**Legend:** — = unimplemented location, read as '0'. Shaded cells are not used by interrupt-on-change.

## 19.0 PERIPHERAL MODULE DISABLE (PMD)

Sleep, Idle and Doze modes allow users to substantially reduce power consumption by slowing or stopping the CPU clock. Even so, peripheral modules still remain clocked, and thus, consume some amount of power. There may be cases where the application needs what these modes do not provide: the ability to allocate limited power resources to the CPU while eliminating power consumption from the peripherals.

The PIC18(L)F25/26K83 family addresses this requirement by allowing peripheral modules to be selectively enabled or disabled, placing them into the lowest possible power mode.

All modules are ON by default following any Reset.

### 19.1 Disabling a Module

Disabling a module has the following effects:

- All clock and control inputs to the module are suspended; there are no logic transitions, and the module will not function.
- The module is held in Reset.
- Any SFR becomes “unimplemented”
  - Writing is disabled
  - Reading returns 00h
- I/O functionality is prioritized as per [Section 16.2, I/O Priorities](#)
- All associated Input Selection registers are also disabled

### 19.2 Enabling a Module

When the PMD register bit is cleared, the module is re-enabled and will be in its Reset state (Power-on Reset). SFR data will reflect the POR Reset values.

Depending on the module, it may take up to one full instruction cycle for the module to become active. There should be no interaction with the module (e.g., writing to registers) for at least one instruction after it has been re-enabled.

### 19.3 Effects of a Reset

Following any Reset, each control bit is set to ‘0’, enabling all modules.

### 19.4 System Clock Disable

Setting SYSCMD (PMD0, [Register 19-1](#)) disables the system clock (FOSC) distribution network to the peripherals. Not all peripherals make use of SYSCLK, so not all peripherals are affected. Refer to the specific peripheral description to see if it will be affected by this bit.

## 19.5 Register Definitions: Peripheral Module Disable

**REGISTER 19-1: PMD0: PMD CONTROL REGISTER 0**

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| SYSCMD  | FVRMD   | HLVDMD  | CRCMD   | SCANMD  | NVMMD   | CLKRMD  | IOCMD   |
| 7       |         |         |         |         |         |         | 0       |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

q = Value depends on condition

- bit 7 **SYSCMD**: Disable Peripheral System Clock Network bit<sup>(1)</sup>  
See description in [Section 19.4 "System Clock Disable"](#).  
1 = System clock network disabled (Fosc)  
0 = System clock network enabled
- bit 6 **FVRMD**: Disable Fixed Voltage Reference bit  
1 = FVR module disabled  
0 = FVR module enabled
- bit 5 **HLVDMD**: Disable Low-Voltage Detect bit  
1 = HLVD module disabled  
0 = HLVD module enabled
- bit 4 **CRCMD**: Disable CRC Engine bit  
1 = CRC module disabled  
0 = CRC module enabled
- bit 3 **SCANMD**: Disable NVM Memory Scanner bit<sup>(2)</sup>  
1 = NVM Memory Scan module disabled  
0 = NVM Memory Scan module enabled
- bit 2 **NVMMD**: NVM Module Disable bit<sup>(3)</sup>  
1 = All Memory reading and writing is disabled; NVMCON registers cannot be written  
0 = NVM module enabled
- bit 1 **CLKRMD**: Disable Clock Reference bit  
1 = CLKR module disabled  
0 = CLKR module enabled
- bit 0 **IOCMD**: Disable Interrupt-on-Change bit, All Ports  
1 = IOC module(s) disabled  
0 = IOC module(s) enabled

**Note 1:** Clearing the SYSCMD bit disables the system clock (Fosc) to peripherals, however peripherals clocked by Fosc/4 are not affected.

**2:** Subject to SCANE bit in CONFIG4H.

**3:** When enabling NVM, a delay of up to 1  $\mu$ s may be required before accessing data.

## REGISTER 19-2: PMD1: PMD CONTROL REGISTER 1

|         |         |         |         |         |         |         |         |
|---------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| NCO1MD  | TMR6MD  | TMR5MD  | TMR4MD  | TMR3MD  | TMR2MD  | TMR1MD  | TMR0MD  |
| bit 7   |         |         |         |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

q = Value depends on condition

|       |                                                                                               |
|-------|-----------------------------------------------------------------------------------------------|
| bit 7 | <b>NCO1MD:</b> Disable NCO1 Module bit<br>1 = NCO1 module disabled<br>0 = NCO1 module enabled |
| bit 6 | <b>TMR6MD:</b> Disable Timer TMR6 bit<br>1 = TMR6 module disabled<br>0 = TMR6 module enabled  |
| bit 5 | <b>TMR5MD:</b> Disable Timer TMR5 bit<br>1 = TMR5 module disabled<br>0 = TMR5 module enabled  |
| bit 4 | <b>TMR4MD:</b> Disable Timer TMR4 bit<br>1 = TMR4 module disabled<br>0 = TMR4 module enabled  |
| bit 3 | <b>TMR3MD:</b> Disable Timer TMR3 bit<br>1 = TMR3 module disabled<br>0 = TMR3 module enabled  |
| bit 2 | <b>TMR2MD:</b> Disable Timer TMR2 bit<br>1 = TMR2 module disabled<br>0 = TMR2 module enabled  |
| bit 1 | <b>TMR1MD:</b> Disable Timer TMR1 bit<br>1 = TMR1 module disabled<br>0 = TMR1 module enabled  |
| bit 0 | <b>TMR0MD:</b> Disable Timer TMR0 bit<br>1 = TMR0 module disabled<br>0 = TMR0 module enabled  |

**REGISTER 19-3: PMD2: PMD CONTROL REGISTER 2**

| U-0   | R/W-0/0 | R/W-0/0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0              |
|-------|---------|---------|-----|-----|---------|---------|----------------------|
| —     | DACMD   | ADCMD   | —   | —   | CMP2MD  | CMP1MD  | ZCDMD <sup>(1)</sup> |
| bit 7 |         |         |     |     |         |         | bit 0                |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

q = Value depends on condition

- bit 7      **Unimplemented:** Read as '0'
- bit 6      **DACMD:** Disable DAC bit
  - 1 = DAC module disabled
  - 0 = DAC module enabled
- bit 5      **ADCMD:** Disable ADCC bit
  - 1 = ADCC module disabled
  - 0 = ADCC module enabled
- bit 4-3    **Unimplemented:** Read as '0'
- bit 2      **CMP2MD:** Disable Comparator CMP2 bit
  - 1 = CMP2 module disabled
  - 0 = CMP2 module enabled
- bit 1      **CMP1MD:** Disable Comparator CMP1 bit
  - 1 = CMP1 module disabled
  - 0 = CMP1 module enabled
- bit 0      **ZCDMD:** Disable Zero-Cross Detect module bit<sup>(1)</sup>
  - 1 = ZCD module disabled
  - 0 = ZCD module enabled

**Note 1:** Subject to  $\overline{\text{ZCD}}$  bit in CONFIG2H.

**REGISTER 19-4: PMD3: PMD CONTROL REGISTER 3**

|         |         |         |         |         |         |         |         |
|---------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| PWM8MD  | PWM7MD  | PWM6MD  | PWM5MD  | CCP4MD  | CCP3MD  | CCP2MD  | CCP1MD  |
| bit 7   |         |         |         |         |         |         | bit 0   |

**Legend:**

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared | q = Value depends on condition                        |

|       |                                                                                                              |
|-------|--------------------------------------------------------------------------------------------------------------|
| bit 7 | <b>PWM8MD:</b> Disable Pulse-Width Modulator PWM8 bit<br>1 = PWM8 module disabled<br>0 = PWM8 module enabled |
| bit 6 | <b>PWM7MD:</b> Disable Pulse-Width Modulator PWM7 bit<br>1 = PWM7 module disabled<br>0 = PWM7 module enabled |
| bit 5 | <b>PWM6MD:</b> Disable Pulse-Width Modulator PWM6 bit<br>1 = PWM6 module disabled<br>0 = PWM6 module enabled |
| bit 4 | <b>PWM5MD:</b> Disable Pulse-Width Modulator PWM5 bit<br>1 = PWM5 module disabled<br>0 = PWM5 module enabled |
| bit 3 | <b>CCP4MD:</b> Disable Capture/Compare/PWM CCP4 bit<br>1 = CCP4 module disabled<br>0 = CCP4 module enabled   |
| bit 2 | <b>CCP3MD:</b> Disable Capture/Compare/PWM CCP3 bit<br>1 = CCP3 module disabled<br>0 = CCP3 module enabled   |
| bit 1 | <b>CCP2MD:</b> Disable Capture/Compare/PWM CCP2 bit<br>1 = CCP2 module disabled<br>0 = CCP2 module enabled   |
| bit 0 | <b>CCP1MD:</b> Disable Capture/Compare/PWM CCP1 bit<br>1 = CCP1 module disabled<br>0 = CCP1 module enabled   |

**REGISTER 19-5: PMD4: PMD CONTROL REGISTER 4**

|         |         |         |     |     |     |     |       |
|---------|---------|---------|-----|-----|-----|-----|-------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | U-0 | U-0 | U-0 | U-0 | U-0   |
| CWG3MD  | CWG2MD  | CWG1MD  | —   | —   | —   | —   | —     |
| bit 7   |         |         |     |     |     |     | bit 0 |

**Legend:**

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared | q = Value depends on condition                        |

- bit 7      **CWG3MD:** Disable CWG3 Module bit  
             1 = CWG3 module disabled  
             0 = CWG3 module enabled
- bit 6      **CWG2MD:** Disable CWG2 Module bit  
             1 = CWG2 module disabled  
             0 = CWG2 module enabled
- bit 5      **CWG1MD:** Disable CWG1 Module bit  
             1 = CWG1 module disabled  
             0 = CWG1 module enabled
- bit 4-0    **Unimplemented:** Read as '0'



## REGISTER 19-6: PMD5: PMD CONTROL REGISTER 5

| U-0   | U-0 | R/W-0/0 | R/W-0/0 | U-0   | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|-------|-----|---------|---------|-------|---------|---------|---------|
| —     | —   | U2MD    | U1MD    | —     | SPI1MD  | I2C2MD  | I2C1MD  |
| bit 7 |     |         |         | bit 0 |         |         |         |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

q = Value depends on condition

|         |                                                                                                                                      |
|---------|--------------------------------------------------------------------------------------------------------------------------------------|
| bit 7-6 | <b>Unimplemented:</b> Read as '0'                                                                                                    |
| bit 5   | <b>U2MD:</b> Disable UART2 bit<br>1 = UART2 module disabled<br>0 = UART2 module enabled                                              |
| bit 4   | <b>U1MD:</b> Disable UART1 bit<br>1 = UART1 module disabled<br>0 = UART1 module enabled                                              |
| bit 3   | <b>Unimplemented:</b> Read as '0'                                                                                                    |
| bit 2   | <b>SPI1MD:</b> Disable SPI1 Module bit<br>1 = SPI1 module disabled<br>0 = SPI1 module enabled                                        |
| bit 1   | <b>I2C2MD:</b> Disable I <sup>2</sup> C2 Module bit<br>1 = I <sup>2</sup> C2 module disabled<br>0 = I <sup>2</sup> C2 module enabled |
| bit 0   | <b>I2C1MD:</b> Disable I <sup>2</sup> C1 Module bit<br>1 = I <sup>2</sup> C1 module disabled<br>0 = I <sup>2</sup> C1 module enabled |

**REGISTER 19-7: PMD6: PMD CONTROL REGISTER 6**

|       |         |         |         |         |         |         |         |
|-------|---------|---------|---------|---------|---------|---------|---------|
| U-0   | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| —     | SMT2MD  | SMT1MD  | CLC4MD  | CLC3MD  | CLC2MD  | CLC1MD  | DSMMD   |
| bit 7 |         |         |         |         |         |         | bit 0   |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

q = Value depends on condition

|       |                                                                                                      |
|-------|------------------------------------------------------------------------------------------------------|
| bit 7 | <b>Unimplemented:</b> Read as '0'                                                                    |
| bit 6 | <b>SMT2MD:</b> Disable SMT2 Module bit<br>1 = SMT2 module disabled<br>0 = SMT2 module enabled        |
| bit 5 | <b>SMT1MD:</b> Disable SMT1 Module bit<br>1 = SMT1 module disabled<br>0 = SMT1 module enabled        |
| bit 4 | <b>CLC1MD:</b> Disable CLC4 Module bit<br>1 = CLC4 module disabled<br>0 = CLC4 module enabled        |
| bit 3 | <b>CLC3MD:</b> Disable CLC3 Module bit<br>1 = CLC3 module disabled<br>0 = CLC3 module enabled        |
| bit 2 | <b>CLC2MD:</b> Disable CLC2 Module bit<br>1 = CLC2 module disabled<br>0 = CLC2 module enabled        |
| bit 1 | <b>CLC1MD:</b> Disable CLC1 Module bit<br>1 = CLC1 module disabled<br>0 = CLC1 module enabled        |
| bit 0 | <b>DSMMD:</b> Disable Data Signal Modulator bit<br>1 = DSM module disabled<br>0 = DSM module enabled |

**REGISTER 19-8: PMD7: PMD CONTROL REGISTER 7**

|         |     |     |     |     |     |         |         |
|---------|-----|-----|-----|-----|-----|---------|---------|
| R/W-0/0 | U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 |
| CANMD   | —   | —   | —   | —   | —   | DMA2MD  | DMA1MD  |
| bit 7   |     |     |     |     |     | bit 0   |         |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

q = Value depends on condition

- bit 7      **CANMD:** Disable CAN Module bit  
             1 = CAN module disabled  
             0 = CAN module enabled
- bit 6-2    **Unimplemented:** Read as '0'
- bit 1      **DMA2MD:** Disable DMA2 Module bit  
             1 = DMA2 module disabled  
             0 = DMA2 module enabled
- bit 0      **DMA1MD:** Disable DMA1 Module bit  
             1 = DMA1 module disabled  
             0 = DMA1 module enabled

**TABLE 19-1: SUMMARY OF REGISTERS ASSOCIATED WITH PERIPHERAL MODULE DISABLE**

| Name | Bit 7  | Bit 6  | Bit 5  | Bit 4  | Bit 3  | Bit 2  | Bit 1  | Bit 0  | Register on Page    |
|------|--------|--------|--------|--------|--------|--------|--------|--------|---------------------|
| PMD0 | SYSCMD | FVRMD  | HLVDMD | CRCMD  | SCANMD | NVMMD  | CLKRMD | IOCMD  | <a href="#">276</a> |
| PMD1 | NCO1MD | TMR6MD | TMR5MD | TMR4MD | TMR3MD | TMR2MD | TMR1MD | TMR0MD | <a href="#">277</a> |
| PMD2 | —      | DACMD  | ADCMD  | —      | —      | CMP2MD | CMP1MD | ZCDMD  | <a href="#">278</a> |
| PMD3 | PWM8MD | PWM7MD | PWM6MD | PWM5MD | CCP4MD | CCP3MD | CCP2MD | CCP1MD | <a href="#">279</a> |
| PMD4 | CWG3MD | CWG2MD | CWG1MD | —      | —      | —      | —      | —      | <a href="#">280</a> |
| PMD5 | —      | —      | U2MD   | U1MD   | —      | SPI1MD | I2C2MD | I2C1MD | <a href="#">281</a> |
| PMD6 | —      | SMT2MD | SMT1MD | CLC4MD | CLC3MD | CLC2MD | CLC1MD | DSMMD  | <a href="#">281</a> |
| PMD7 | CANMD  | —      | —      | —      | —      | —      | DMA2MD | DMA1MD | <a href="#">283</a> |

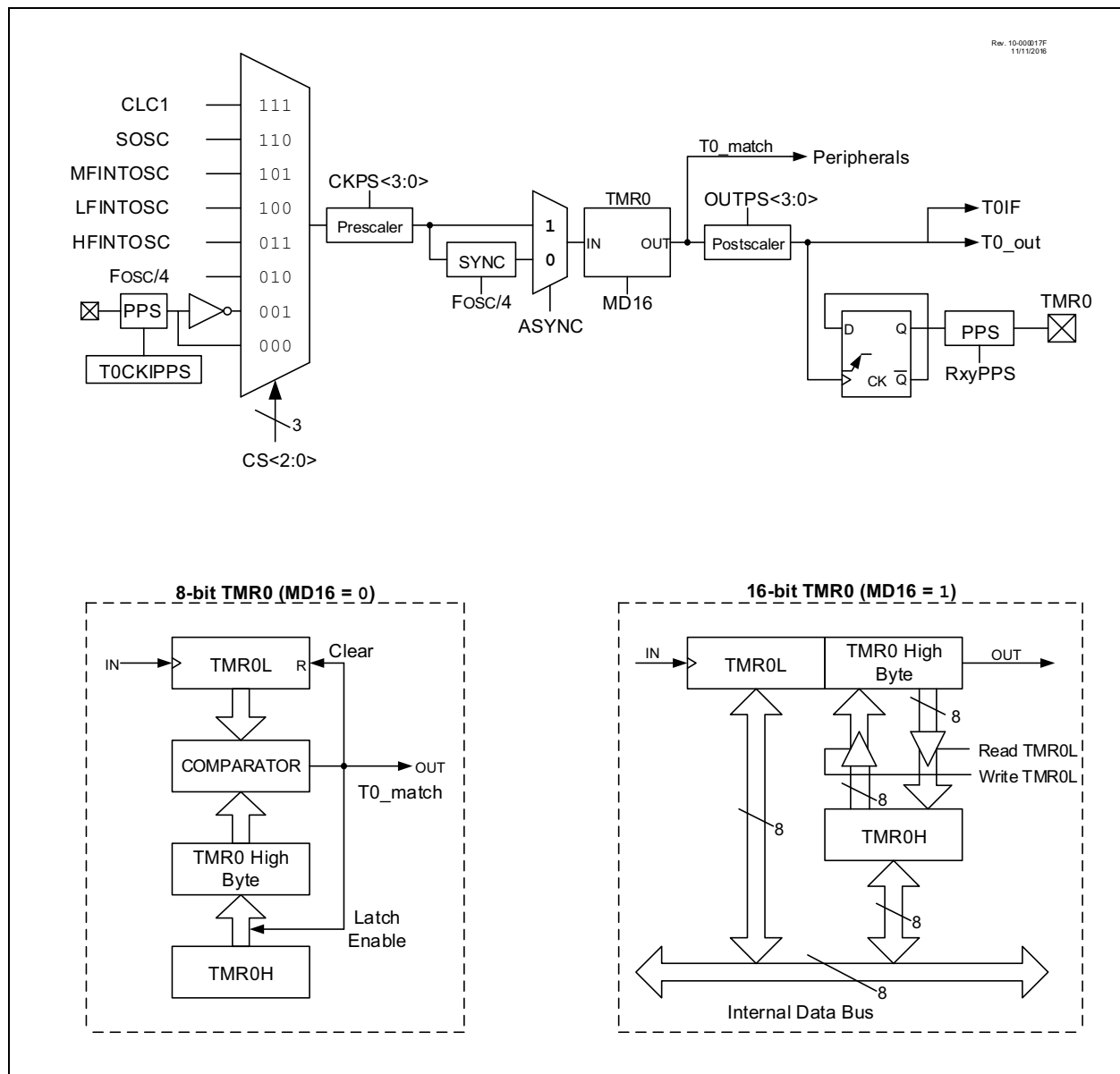
**Legend:** — = unimplemented location, read as '0'. Shaded cells are not used by peripheral module disable.

## 20.0 TIMER0 MODULE

Timer0 module is an 8/16-bit timer/counter with the following features:

- 16-bit timer/counter
- 8-bit timer/counter with programmable period
- Synchronous or asynchronous operation
- Selectable clock sources
- Programmable prescaler
- Programmable postscaler
- Operation during Sleep mode
- Interrupt on match or overflow
- Output on I/O pin (via PPS) or to other peripherals

**FIGURE 20-1: BLOCK DIAGRAM OF TIMER0**



## 20.1 Timer0 Operation

Timer0 can operate as either an 8-bit timer/counter or a 16-bit timer/counter. The mode is selected with the MD16 bit of the T0CON register.

### 20.1.1 16-BIT MODE

The register pair TMR0H:TMR0L increments on the rising edge of the clock source. A 15-bit prescaler on the clock input gives several prescale options (see prescaler control bits, CKPS<3:0> in the T0CON1 register).

#### 20.1.1.1 Timer0 Reads and Writes in 16-Bit Mode

In 16-bit mode, in order to avoid rollover between reading high and low registers, the TMR0H register is a buffered copy of the actual high byte of Timer0, which is neither directly readable, nor writable (see [Figure 20-1](#)). TMR0H is updated with the contents of the high byte of Timer0 during a read of TMR0L. This provides the ability to read all 16 bits of Timer0 without having to verify that the read of the high and low byte was valid, due to a rollover between successive reads of the high and low byte.

Similarly, a write to the high byte of Timer0 must also take place through the TMR0H Buffer register. The high byte is updated with the contents of TMR0H when a write occurs to TMR0L. This allows all 16 bits of Timer0 to be updated at once.

### 20.1.2 8-BIT MODE

In 8-bit mode, the value of TMR0L is compared to that of the Period buffer, a copy of TMR0H, on each clock cycle. When the two values match, the following events happen:

- TMR0\_out goes high for one prescaled clock period
- TMR0L is reset
- The contents of TMR0H are copied to the period buffer

In 8-bit mode, the TMR0L and TMR0H registers are both directly readable and writable. The TMR0L register is cleared on any device Reset, while the TMR0H register initializes at FFh.

Both the prescaler and postscaler counters are cleared on the following events:

- A write to the TMR0L register
- A write to either the T0CON0 or T0CON1 registers
- Any device Reset – Power-on Reset (POR), MCLR Reset, Watchdog Timer Reset (WDTR) or
- Brown-out Reset (BOR)

### 20.1.3 COUNTER MODE

In Counter mode, the prescaler is normally disabled by setting the CKPS bits of the T0CON1 register to '0000'. Each rising edge of the clock input (or the output of the prescaler if the prescaler is used) increments the counter by '1'.

### 20.1.4 TIMER MODE

In Timer mode, the Timer0 module will increment every instruction cycle as long as there is a valid clock signal and the CKPS bits of the T0CON1 register ([Register 20-2](#)) are set to '0000'. When a prescaler is added, the timer will increment at the rate based on the prescaler value.

### 20.1.5 ASYNCHRONOUS MODE

When the ASYNC bit of the T0CON1 register is set (ASYNC = 1), the counter increments with each rising edge of the input source (or output of the prescaler, if used). Asynchronous mode allows the counter to continue operation during Sleep mode provided that the clock also continues to operate during Sleep.

### 20.1.6 SYNCHRONOUS MODE

When the ASYNC bit of the T0CON1 register is clear (ASYNC = 0), the counter clock is synchronized to the system clock (Fosc/4). When operating in Synchronous mode, the counter clock frequency cannot exceed Fosc/4.

## 20.2 Clock Source Selection

The CS<2:0> bits of the T0CON1 register are used to select the clock source for Timer0. [Register 20-2](#) displays the clock source selections.

### 20.2.1 INTERNAL CLOCK SOURCE

When the internal clock source is selected, Timer0 operates as a timer and will increment on multiples of the clock source, as determined by the Timer0 prescaler.

### 20.2.2 EXTERNAL CLOCK SOURCE

When an external clock source is selected, Timer0 can operate as either a timer or a counter. Timer0 will increment on multiples of the rising edge of the external clock source, as determined by the Timer0 prescaler.

## 20.3 Programmable Prescaler

A software programmable prescaler is available for exclusive use with Timer0. There are 16 prescaler options for Timer0 ranging in powers of two from 1:1 to 1:32768. The prescaler values are selected using the CKPS<3:0> bits of the T0CON1 register.

The prescaler is not directly readable or writable. Clearing the prescaler register can be done by writing to the TMR0L register or to the T0CON0/T0CON1 register or by any Reset.

## 20.4 Programmable Postscaler

A software programmable postscaler (output divider) is available for exclusive use with Timer0. There are 16 postscaler options for Timer0 ranging from 1:1 to 1:16. The postscaler values are selected using the OUTPS bits of the T0CON0 register.

The postscaler is not directly readable or writable. Clearing the postscaler register can be done by writing to the TMR0L register or to the T0CON0/T0CON1 register or by any Reset.

## 20.5 Operation During Sleep

When operating synchronously, Timer0 will halt. When operating asynchronously, Timer0 will continue to increment and wake the device from Sleep (if Timer0 interrupts are enabled) provided that the input clock source is active.

## 20.6 Timer0 Interrupts

The Timer0 interrupt flag bit (TMR0IF) is set when either of the following conditions occur:

- 8-bit TMR0L matches the TMR0H value
- 16-bit TMR0 rolls over from 'FFFFh'

When the postscaler bits (OUTPS) are set to 1:1 operation (no division), the T0IF flag bit will be set with every TMR0 match or rollover. In general, the TMR0IF flag bit will be set every OUTPS + 1 matches or rollovers.

If Timer0 interrupts are enabled (TMR0IE bit of the PIE3 register = '1'), the CPU will be interrupted and the device may wake from Sleep (see [Section 20.2 "Clock Source Selection"](#) for more details).

## 20.7 Timer0 Output

The Timer0 output can be routed to any I/O pin via the RxyPPS output selection register (see [Section 17.0 "Peripheral Pin Select \(PPS\) Module"](#) for additional information). The Timer0 output can also be used by other peripherals, such as the auto-conversion trigger of the Analog-to-Digital Converter. Finally, the Timer0 output can be monitored through software via the Timer0 output bit (OUT) of the T0CON0 register ([Register 20-1](#)).

TMR0\_out will be a pulse of one postscaled clock period when a match occurs between TMR0L and PR0 (Period register for TMR0) in 8-bit mode, or when TMR0 rolls over in 16-bit mode. The Timer0 output is a 50% duty cycle that toggles on each TMR0\_out rising clock edge.

## 20.8 Register Definitions: Timer0 Control

### REGISTER 20-1: T0CON0: TIMER0 CONTROL REGISTER 0

| R/W-0/0 | U-0 | R-0 | R/W-0/0 | R/W-0/0    | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|-----|-----|---------|------------|---------|---------|---------|
| EN      | —   | OUT | MD16    | OUTPS<3:0> |         |         |         |
| bit 7   |     |     |         |            |         |         | bit 0   |

#### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

|         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| bit 7   | <b>EN:</b> TMR0 Enable bit<br>1 = The module is enabled and operating<br>0 = The module is disabled and in the lowest power mode                                                                                                                                                                                                                                                                                                                                                             |
| bit 6   | <b>Unimplemented:</b> Read as '0'                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| bit 5   | <b>OUT:</b> TMR0 Output bit (read-only)<br>TMR0 output bit                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| bit 4   | <b>MD16:</b> TMR0 Operating as 16-Bit Timer Select bit<br>1 = TMR0 is a 16-bit timer<br>0 = TMR0 is an 8-bit timer                                                                                                                                                                                                                                                                                                                                                                           |
| bit 3-0 | <b>OUTPS&lt;3:0&gt;:</b> TMR0 Output Postscaler (Divider) Select bits<br>1111 = 1:16 Postscaler<br>1110 = 1:15 Postscaler<br>1101 = 1:14 Postscaler<br>1100 = 1:13 Postscaler<br>1011 = 1:12 Postscaler<br>1010 = 1:11 Postscaler<br>1001 = 1:10 Postscaler<br>1000 = 1:9 Postscaler<br>0111 = 1:8 Postscaler<br>0110 = 1:7 Postscaler<br>0101 = 1:6 Postscaler<br>0100 = 1:5 Postscaler<br>0011 = 1:4 Postscaler<br>0010 = 1:3 Postscaler<br>0001 = 1:2 Postscaler<br>0000 = 1:1 Postscaler |

# PIC18(L)F25/26K83

## REGISTER 20-2: T0CON1: TIMER0 CONTROL REGISTER 1

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0   | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|---------|---------|---------|-----------|---------|---------|---------|
| CS<2:0> |         |         | ASYNC   | CKPS<3:0> |         |         |         |
| bit 7   |         |         |         |           |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-5

**CS<2:0>**: Timer0 Clock Source Select bits

111 = CLC1

110 = SOSC

101 = MFINTOSC (500 kHz)

100 = LFINTOSC

011 = HFINTOSC

010 = Fosc/4

001 = Pin selected by T0CKIPPS (Inverted)

000 = Pin selected by T0CKIPPS (Non-inverted)

bit 4

**ASYNC**: TMR0 Input Asynchronization Enable bit

1 = The input to the TMR0 counter is not synchronized to system clocks

0 = The input to the TMR0 counter is synchronized to Fosc/4

bit 3-0

**CKPS<3:0>**: Prescaler Rate Select bit

1111 = 1:32768

1110 = 1:16384

1101 = 1:8192

1100 = 1:4096

1011 = 1:2048

1010 = 1:1024

1001 = 1:512

1000 = 1:256

0111 = 1:128

0110 = 1:64

0101 = 1:32

0100 = 1:16

0011 = 1:8

0010 = 1:4

0001 = 1:2

0000 = 1:1



## REGISTER 20-3: TMR0L: TIMER0 COUNT REGISTER

|            |         |         |         |         |         |         |         |
|------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0    | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| TMR0L<7:0> |         |         |         |         |         |         |         |
| bit 7      |         |         |         | bit 0   |         |         |         |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0 **TMR0L<7:0>**: TMR0 Counter bits <7:0>

## REGISTER 20-4: TMR0H: TIMER0 PERIOD REGISTER

|             |         |         |         |         |         |         |         |
|-------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-1/1     | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
| TMR0H<15:8> |         |         |         |         |         |         |         |
| bit 7       |         |         |         | bit 0   |         |         |         |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0 When MD16 = 0  
**PR0<7:0>**: TMR0 Period Register Bits <7:0>  
 When MD16 = 1  
**TMR0H<15:8>**: TMR0 Counter bits <15:8>

## TABLE 20-1: SUMMARY OF REGISTERS ASSOCIATED WITH TIMER0

| Name   | Bit 7       | Bit 6 | Bit 5 | Bit 4 | Bit 3      | Bit 2 | Bit 1 | Bit 0 | Register on Page    |
|--------|-------------|-------|-------|-------|------------|-------|-------|-------|---------------------|
| T0CON0 | EN          | —     | OUT   | MD16  | OUTPS<3:0> |       |       |       | <a href="#">287</a> |
| T0CON1 | CS<2:0>     |       |       | ASYNC | CKPS<3:0>  |       |       |       | <a href="#">288</a> |
| TMR0L  | TMR0L<7:0>  |       |       |       |            |       |       |       | <a href="#">289</a> |
| TMR0H  | TMR0H<15:8> |       |       |       |            |       |       |       | <a href="#">289</a> |

**Legend:** — = unimplemented location, read as '0'. Shaded cells are not used by Timer0.



## 21.1 Timer1/3/5 Operation

The Timer1/3/5 module is a 16-bit incrementing counter which is accessed through the TMRxH:TMRxL register pair. Writes to TMRxH or TMRxL directly update the counter.

When used with an internal clock source, the module is a timer and increments on every instruction cycle. When used with an external clock source, the module can be used as either a timer or counter and increments on every selected edge of the external source.

Timer1/3/5 is enabled by configuring the ON and GE bits in the TxCON and TxGCON registers, respectively. [Table 21-1](#) displays the Timer1/3/5 enable selections.

**TABLE 21-1: TIMER1/3/5 ENABLE SELECTIONS**

| ON | GE | Timer1/3/5 Operation |
|----|----|----------------------|
| 1  | 1  | Count Enabled        |
| 1  | 0  | Always On            |
| 0  | 1  | Off                  |
| 0  | 0  | Off                  |

## 21.2 Clock Source Selection

The CS<4:0> bits of the TMRxCLK register ([Register 21-3](#)) are used to select the clock source for Timer1/3/5. The five TMRxCLK bits allow the selection of several possible synchronous and asynchronous clock sources. [Register 21-3](#) displays the clock source selections.

### 21.2.1 INTERNAL CLOCK SOURCE

When the internal clock source is selected the TMRxH:TMRxL register pair will increment on multiples of FOSC as determined by the Timer1/3/5 prescaler.

When the FOSC internal clock source is selected, the Timer1/3/5 register value will increment by four counts every instruction clock cycle. Due to this condition, a 2 LSB error in resolution will occur when reading the Timer1/3/5 value. To utilize the full resolution of Timer1/3/5, an asynchronous input signal must be used to gate the Timer1/3/5 clock input.

The following asynchronous sources may be used at the Timer1/3/5 gate:

- Asynchronous event on the TxGPPS pin
- TMR0OUT
- TMR1/3/5OUT (excluding the TMR for which it is being used)
- TMR 2/4/6OUT (post-scaled)
- CMP1/2OUT
- SMT1\_match
- NCO1OUT
- PWM3/4 OUT
- CCP1/2/3/4 OUT
- CLC1/2/3/4 OUT
- ZCDOUT

**Note:** In Counter mode, a falling edge must be registered by the counter prior to the first incrementing rising edge after any one or more of the following conditions:

- Timer1/3/5 enabled after POR
- Write to TMRxH or TMRxL
- Timer1/3/5 is disabled
- Timer1/3/5 is disabled (TMRxON = 0) when TxCKI is high then Timer1/3/5 is enabled (TMRxON = 1) when TxCKI is low.

### 21.2.2 EXTERNAL CLOCK SOURCE

When the external clock source is selected, the Timer1/3/5 module may work as a timer or a counter.

When enabled to count, Timer1/3/5 is incremented on the rising edge of the external clock input of the TxCKIPPS pin. This external clock source can be synchronized to the microcontroller system clock or it can run asynchronously.

When used as a timer with a clock oscillator, an external 32.768 kHz crystal can be used in conjunction with the dedicated secondary internal oscillator circuit.

## 21.3 Timer1/3/5 Prescaler

Timer1/3/5 has four prescaler options allowing 1, 2, 4 or 8 divisions of the clock input. The CKPS bits of the TxCON register control the prescale counter. The prescale counter is not directly readable or writable; however, the prescaler counter is cleared upon a write to TMRxH or TMRxL.

## 21.4 Timer1/3/5 Operation in Asynchronous Counter Mode

If control bit  $\overline{\text{SYNC}}$  of the TxCON register is set, the external clock input is not synchronized. The timer increments asynchronously to the internal phase clocks. If external clock source is selected then the timer will continue to run during Sleep and can generate an interrupt on overflow, which will wake up the processor. However, special precautions in software are needed to read/write the timer (see [Section 21.4.1 “Reading and Writing Timer1/3/5 in Asynchronous Counter Mode”](#)).

**Note:** When switching from synchronous to asynchronous operation, it is possible to skip an increment. When switching from asynchronous to synchronous operation, it is possible to produce an additional increment.

### 21.4.1 READING AND WRITING TIMER1/3/5 IN ASYNCHRONOUS COUNTER MODE

Reading TMRxH or TMRxL while the timer is running from an external asynchronous clock will ensure a valid read (taken care of in hardware). However, the user should keep in mind that reading the 16-bit timer in two 8-bit values itself, poses certain problems, since the timer may overflow between the reads. For writes, it is recommended that the user simply stop the timer and write the desired values. A write contention may occur by writing to the timer registers, while the register is incrementing. This may produce an unpredictable value in the TMRxH:TMRxL register pair.

## 21.5 Timer1/3/5 16-Bit Read/Write Mode

Timer1/3/5 can be configured to read and write all 16 bits of data, to and from, the 8-bit TMRxL and TMRxH registers, simultaneously. The 16-bit read and write operations are enabled by setting the RD16 bit of the TxCON register.

To accomplish this function, the TMRxH register value is mapped to a buffer register called the TMRxH buffer register. While in 16-Bit mode, the TMRxH register is not directly readable or writable and all read and write operations take place through the use of this TMRxH buffer register.

When a read from the TMRxL register is requested, the value of the TMRxH register is simultaneously loaded into the TMRxH buffer register. When a read from the TMRxH register is requested, the value is provided from the TMRxH buffer register instead. This provides the user with the ability to accurately read all 16 bits of the Timer1/3/5 value from a single instance in time. Reference the block diagram in [Figure 21-2](#) for more details.

In contrast, when not in 16-Bit mode, the user must read each register separately and determine if the values have become invalid due to a rollover that may have occurred between the read operations.

When a write request of the TMRxL register is requested, the TMRxH buffer register is simultaneously updated with the contents of the TMRxH register. The value of TMRxH must be preloaded into the TMRxH buffer register prior to the write request for the TMRxL register. This provides the user with the ability to write all 16 bits to the TMRxL:TMRxH register pair at the same time.

Any requests to write to the TMRxH directly does not clear the Timer1/3/5 prescaler value. The prescaler value is only cleared through write requests to the TMRxL register.



## 21.6.2 TIMER1/3/5 GATE SOURCE SELECTION

The gate source for Timer1/3/5 can be selected using the GSS<4:0> bits of the TMRxGATE register ([Register 21-4](#)). The polarity selection for the gate source is controlled by the TxGPOL bit of the TxGCON register ([Register 21-2](#)).

Any of the above mentioned signals can be used to trigger the gate. The output of the CMPx can be synchronized to the Timer1/3/5 clock or left asynchronous. For more information see [Section 39.3.1 “Comparator Output Synchronization”](#).

## 21.6.3 TIMER1/3/5 GATE TOGGLE MODE

When Timer1/3/5 Gate Toggle mode is enabled, it is possible to measure the duration between every rising and falling edge of the gate signal.

The Timer1/3/5 gate source is routed through a flip-flop that changes state on every incrementing edge of the signal. See [Figure 21-5](#) for timing details.

Timer1/3/5 Gate Toggle mode is enabled by setting the GTM bit of the TxGCON register. When the GTM bit is cleared, the flip-flop is cleared and held clear. This is necessary in order to control which edge is measured.

**Note:** Enabling Toggle mode at the same time as changing the gate polarity may result in indeterminate operation.

## 21.6.4 TIMER1/3/5 GATE SINGLE-PULSE MODE

When Timer1/3/5 Gate Single-Pulse mode is enabled, it is possible to capture a single-pulse gate event. Timer1/3/5 Gate Single-Pulse mode is first enabled by setting the GSPM bit in the TxGCON register. Next, the GGO/DONE bit in the TxGCON register must be set. The Timer1/3/5 will be fully enabled on the next incrementing edge of the gate signal. On the next trailing edge of the pulse, the GGO/DONE bit will automatically be cleared. No other gate events will be allowed to increment Timer1/3/5 until the GGO/DONE bit is once again set in software.

Clearing the TxGSPM bit of the TxGCON register will also clear the GGO/DONE bit. See [Figure 21-6](#) for timing details.

Enabling the Toggle mode and the Single-Pulse mode simultaneously will permit both sections to work together. This allows the period on the Timer1/3/5 gate source to be measured. See [Figure 21-7](#) for timing details.

## 21.6.5 TIMER1/3/5 GATE VALUE STATUS

When Timer1/3/5 Gate Value Status is utilized, it is possible to read the most current level of the gate signal. The value is stored in the GVAL bit in the TxGCON register. The GVAL bit is valid even when the Timer1/3/5 gate is not enabled (GE bit is cleared).

## 21.6.6 TIMER1/3/5 GATE EVENT INTERRUPT

When Timer1/3/5 Gate Event Interrupt is enabled, it is possible to generate an interrupt upon the completion of a gate event. When the falling edge of GVAL occurs, the TMRxGIF flag bit in the respective PIR register will be set. If the TMRxGIE bit in the respective PIE register is set, then an interrupt will be recognized.

The TMRxGIF flag bit operates even when the Timer1/3/5 gate is not enabled (GE bit is cleared).

For more information on selecting high or low priority status for the Timer1/3/5 Gate Event Interrupt see [Section 9.0 “Interrupt Controller”](#).

## 21.7 Timer1/3/5 Interrupt

The Timer1/3/5 register pair (TMRxH:TMRxL) increments to FFFFh and rolls over to 0000h. When Timer1/3/5 rolls over, the Timer1/3/5 interrupt flag bit of the respective PIR register is set. To enable the interrupt-on-rollover, you must set these bits:

- ON bit of the TxCON register
- TMRxIE bits of the respective PIE register
- GIE/GIEH bit of the INTCON0 register

The interrupt is cleared by clearing the TMRxIF bit in the Interrupt Service Routine.

For more information on selecting high or low priority status for the Timer1/3/5 Overflow Interrupt, see [Section 9.0 “Interrupt Controller”](#).

**Note:** The TMRxH:TMRxL register pair and the TMRxIF bit should be cleared before enabling interrupts.

## 21.8 Timer1/3/5 Operation During Sleep

Timer1/3/5 can only operate during Sleep when set up in Asynchronous Counter mode. In this mode, an external crystal or clock source can be used to increment the counter. To set up the timer to wake the device:

- ON bit of the TxCON register must be set
- TMRxIE bit of the respective PIE register must be set
- $\overline{\text{SYNC}}$  bit of the TxCON register must be set
- Configure the TMRxCLK register for using secondary oscillator as the clock source
- Enable the SOSSEN bit of the OSCEN register ([Register 7-7](#))

The device will wake-up on an overflow and execute the next instruction. If the GIE/GIEH bit of the INTCON0 register is set, the device will call the Interrupt Service Routine.

The secondary oscillator will continue to operate in Sleep regardless of the SYNC bit setting.

## 21.9 CCP Capture/Compare Time Base

The CCP modules use the TMRxH:TMRxL register pair as the time base when operating in Capture or Compare mode.

In Capture mode, the value in the TMRxH:TMRxL register pair is copied into the CCPRxH:CCPRxL register pair on a configured event.

In Compare mode, an event is triggered when the value in the CCPRxH:CCPRxL register pair matches the value in the TMRxH:TMRxL register pair. This event can be a Special Event Trigger.

For more information, see [Section 23.0 “Capture/Compare/PWM Module”](#).

## 21.10 CCP Special Event Trigger

When any of the CCP's are configured to trigger a special event, the trigger will clear the TMRxH:TMRxL register pair. This special event does not cause a Timer1/3/5 interrupt. The CCP module may still be configured to generate a CCP interrupt.

In this mode of operation, the CCPRxH:CCPRxL register pair becomes the period register for Timer1/3/5.

Timer1/3/5 should be synchronized and Fosc/4 should be selected as the clock source in order to utilize the Special Event Trigger. Asynchronous operation of Timer1/3/5 can cause a Special Event Trigger to be missed.

In the event that a write to TMRxH or TMRxL coincides with a Special Event Trigger from the CCP, the write will take precedence.

**FIGURE 21-3: TIMER1/3/5 INCREMENTING EDGE**

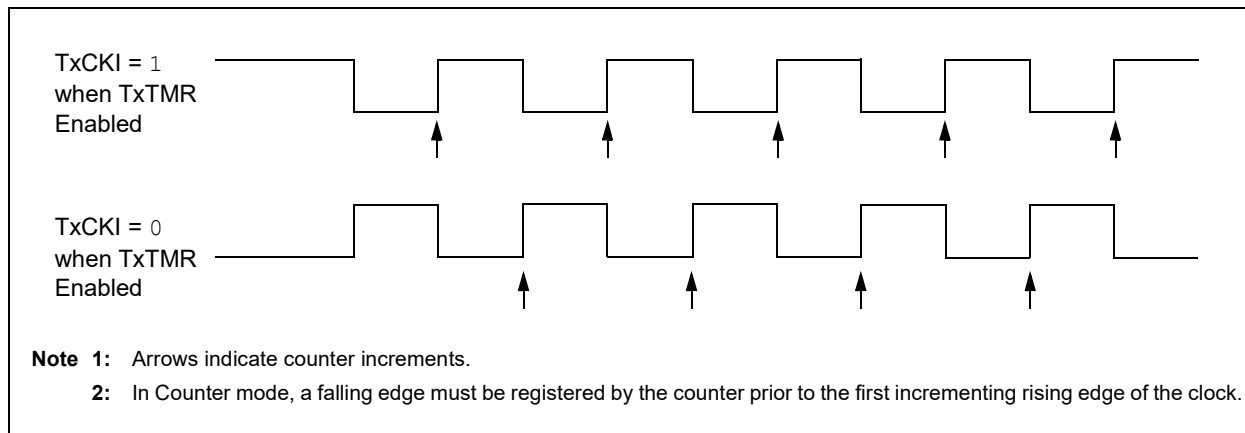


**FIGURE 21-4: TIMER1/3/5 GATE ENABLE MODE**

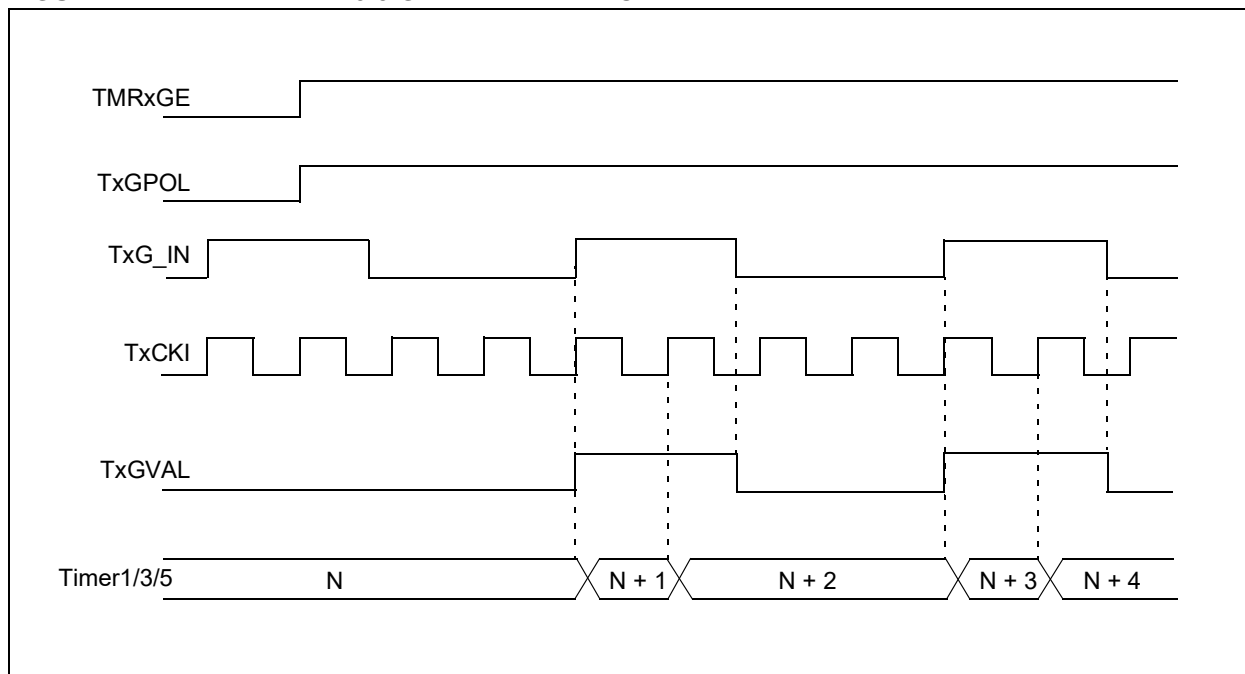




**FIGURE 21-5: TIMER1/3/5 GATE TOGGLE MODE**

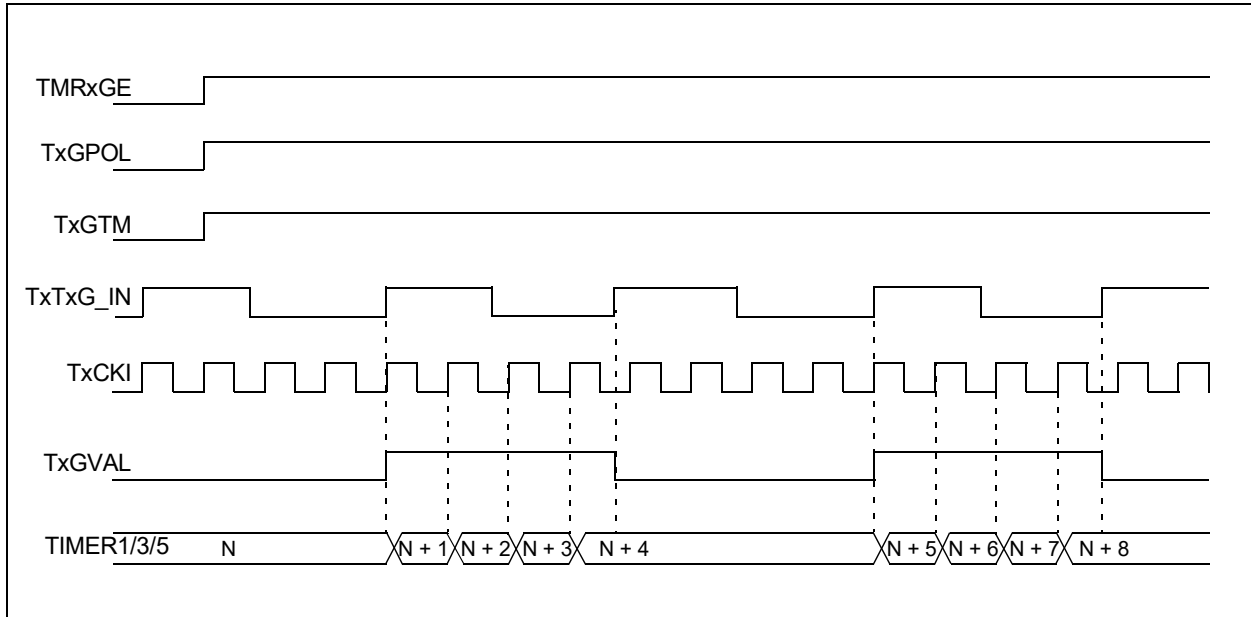


**FIGURE 21-6: TIMER1/3/5 GATE SINGLE-PULSE MODE**

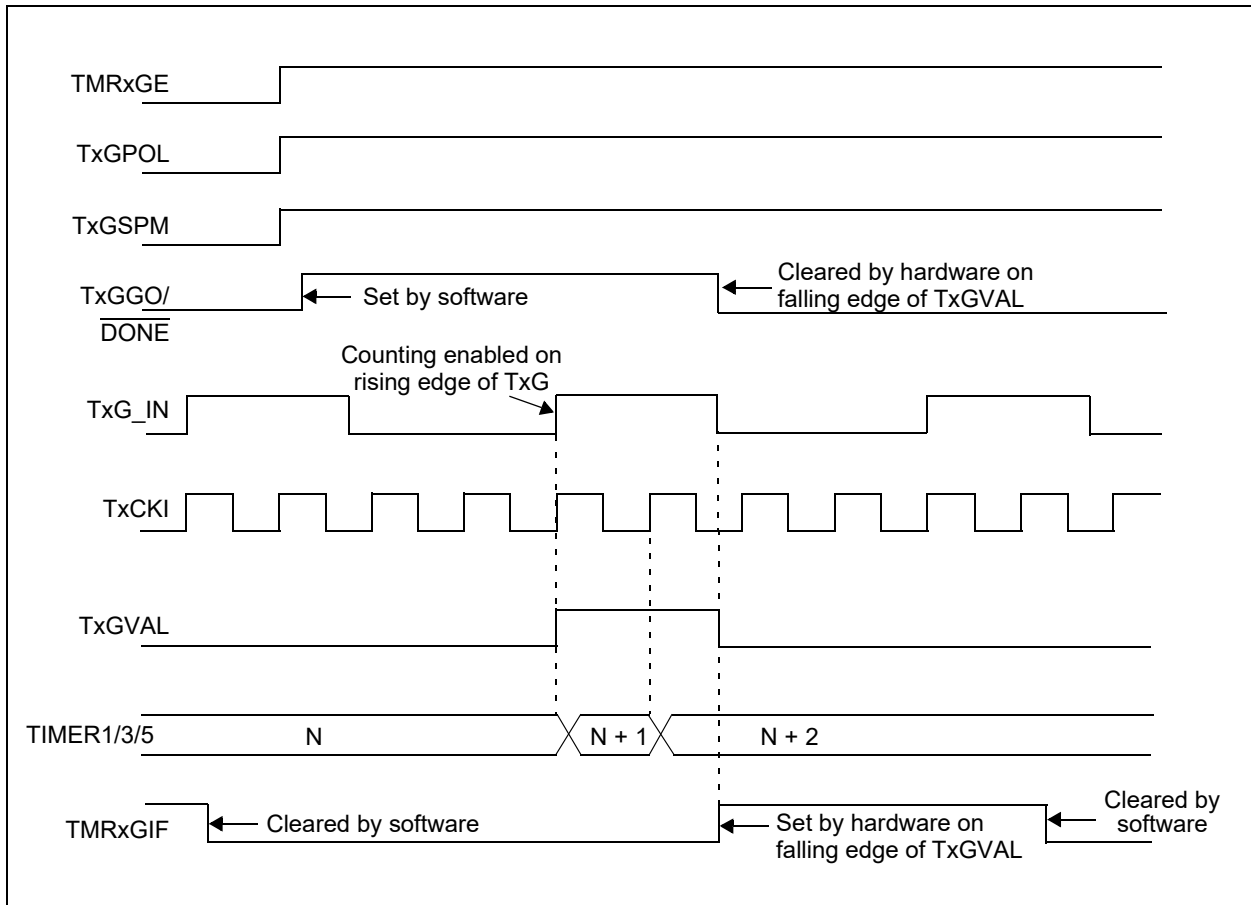
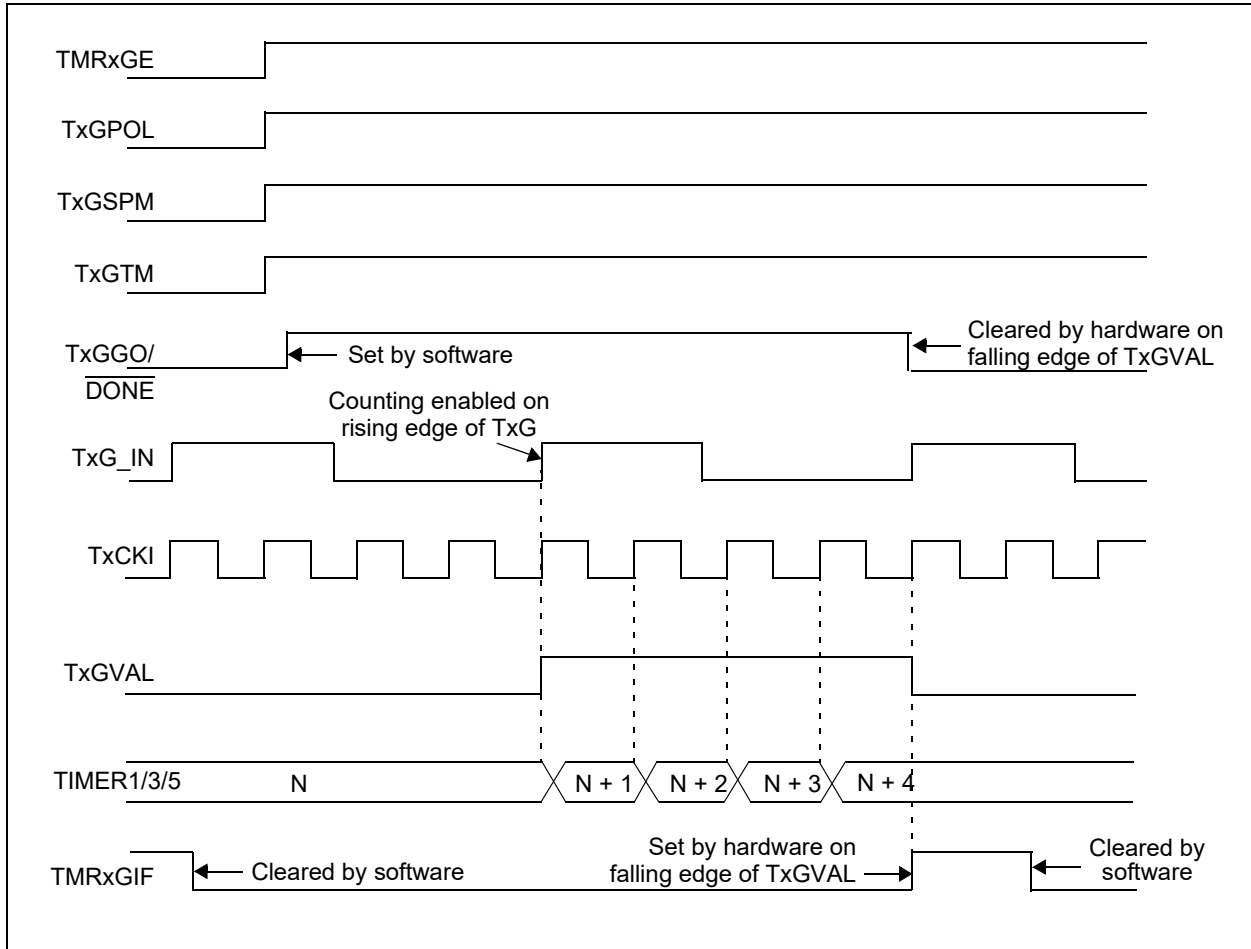


**FIGURE 21-7: TIMER1/3/5 GATE SINGLE-PULSE AND TOGGLE COMBINED MODE**



## 21.11 Peripheral Module Disable

When a peripheral module is not used or inactive, the module can be disabled by setting the Module Disable bit in the PMD registers. This will reduce power consumption to an absolute minimum. Setting the PMD bits holds the module in Reset and disconnects the module's clock source. The Module Disable bits for Timer1 (TMR1MD), Timer3 (TMR3MD) and Timer5 (TMR5MD) are in the respective PMD registers. See [Section 19.0 "Peripheral Module Disable \(PMD\)"](#) for more information.

## 21.12 Register Definitions: Timer1/3/5

Long bit name prefixes for the Timer1/3/5 are shown below. Refer to [Section 1.3.2.2 “Long Bit Names”](#) for more information.

| Peripheral | Bit Name Prefix |
|------------|-----------------|
| Timer1     | T1              |
| Timer3     | T3              |
| Timer5     | T5              |

### REGISTER 21-1: TXCON: TIMERx CONTROL REGISTER

| U-0   | U-0 | R/W-0/u   | R/W-0/u | U-0 | R/W-0/u                  | R/W-0/0 | R/W-0/u |
|-------|-----|-----------|---------|-----|--------------------------|---------|---------|
| —     | —   | CKPS<1:0> |         | —   | $\overline{\text{SYNC}}$ | RD16    | ON      |
| bit 7 |     |           |         |     |                          |         | bit 0   |

#### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as ‘0’

-n = Value at POR

‘1’ = Bit is set

‘0’ = Bit is cleared

u = unchanged

bit 7-6 **Unimplemented:** Read as ‘0’

bit 5-4 **CKPS<1:0>:** Timerx Input Clock Prescale Select bits

11 = 1:8 Prescale value

10 = 1:4 Prescale value

01 = 1:2 Prescale value

00 = 1:1 Prescale value

bit 3 **Unimplemented:** Read as ‘0’

bit 2 **SYNC:** Timerx External Clock Input Synchronization Control bit

TMRxCLK = Fosc/4 or Fosc:

This bit is ignored. Timer1 uses the incoming clock as is.

Else:

1 = Do not synchronize external clock input

0 = Synchronize external clock input with system clock

bit 1 **RD16:** 16-Bit Read/Write Mode Enable bit

1 = Enables register read/write of Timerx in one 16-bit operation

0 = Enables register read/write of Timerx in two 8-bit operation

bit 0 **ON:** Timerx On bit

1 = Enables Timerx

0 = Disables Timerx

## REGISTER 21-2: TxGCON: TIMERx GATE CONTROL REGISTER

| R/W-0/u | R/W-0/u | R/W-0/u | R/W-0/u | R/W-0/u  | R-x  | U-0   | U-0 |
|---------|---------|---------|---------|----------|------|-------|-----|
| GE      | GPOL    | GTM     | GSPM    | GGO/DONE | GVAL | —     | —   |
| bit 7   |         |         |         |          |      | bit 0 |     |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7

**GE:** Timerx Gate Enable bit

If TMRxON = 1:

1 = Timerx counting is controlled by the Timerx gate function

0 = Timerx is always counting

If TMRxON = 0:

This bit is ignored

bit 6

**GPOL:** Timerx Gate Polarity bit

1 = Timerx gate is active-high (Timerx counts when gate is high)

0 = Timerx gate is active-low (Timerx counts when gate is low)

bit 5

**GTM:** Timerx Gate Toggle Mode bit

1 = Timerx Gate Toggle mode is enabled

0 = Timerx Gate Toggle mode is disabled and Toggle flip-flop is cleared

Timerx Gate Flip Flop Toggles on every rising edge

bit 4

**GSPM:** Timerx Gate Single Pulse Mode bit

1 = Timerx Gate Single Pulse mode is enabled and is controlling Timerx gate)

0 = Timerx Gate Single Pulse mode is disabled

bit 3

**GGO/DONE:** Timerx Gate Single Pulse Acquisition Status bit

1 = Timerx Gate Single Pulse Acquisition is ready, waiting for an edge

0 = Timerx Gate Single Pulse Acquisition has completed or has not been started.

This bit is automatically cleared when TxGSPM is cleared.

bit 2

**GVAL:** Timerx Gate Current State bit

Indicates the current state of the Timerx gate that could be provided to TMRxH:TMRxL

Unaffected by Timerx Gate Enable (TMRxGE)

bit 1-0

**Unimplemented:** Read as '0'

## REGISTER 21-3: TxCLK: TIMERx CLOCK REGISTER

|       |     |     |         |         |         |         |         |
|-------|-----|-----|---------|---------|---------|---------|---------|
| U-0   | U-0 | U-0 | R/W-0/u | R/W-0/u | R/W-0/u | R/W-0/u | R/W-0/u |
| —     | —   | —   | CS<4:0> |         |         |         |         |
| bit 7 |     |     |         |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

u = unchanged

bit 7-5

**Unimplemented:** Read as '0'

bit 4-0

**CS<4:0>:** Timerx Clock Source Selection bits

| CS          | Timer1             | Timer3             | Timer5             |
|-------------|--------------------|--------------------|--------------------|
|             | Clock Source       | Clock Source       | Clock Source       |
| 11111-10001 | Reserved           | Reserved           | Reserved           |
| 10000       | CLC4               | CLC4               | CLC4               |
| 01111       | CLC3               | CLC3               | CLC3               |
| 01110       | CLC2               | CLC2               | CLC2               |
| 01101       | CLC1               | CLC1               | CLC1               |
| 01100       | TMR5 overflow      | TMR5 overflow      | Reserved           |
| 01011       | TMR3 overflow      | Reserved           | TMR3 overflow      |
| 01010       | Reserved           | TMR1 overflow      | TMR1 overflow      |
| 01001       | TMR0 overflow      | TMR0 overflow      | TMR0 overflow      |
| 01000       | CLKREF             | CLKREF             | CLKREF             |
| 00111       | SOSC               | SOSC               | SOSC               |
| 00110       | MFINTOSC (32 kHz)  | MFINTOSC (32 kHz)  | MFINTOSC (32 kHz)  |
| 00101       | MFINTOSC (500 kHz) | MFINTOSC (500 kHz) | MFINTOSC (500 kHz) |
| 00100       | LFINTOSC           | LFINTOSC           | LFINTOSC           |
| 00011       | HFINTOSC           | HFINTOSC           | HFINTOSC           |
| 00010       | Fosc               | Fosc               | Fosc               |
| 00001       | Fosc/4             | Fosc/4             | Fosc/4             |
| 00000       | T1CKIPPS           | T3CKIPPS           | T5CKIPPS           |

**REGISTER 21-4: TxGATE: TIMERx GATE ISM REGISTER**

|       |     |     |          |         |         |         |         |
|-------|-----|-----|----------|---------|---------|---------|---------|
| U-0   | U-0 | U-0 | R/W-0/u  | R/W-0/u | R/W-0/u | R/W-0/u | R/W-0/u |
| —     | —   | —   | GSS<4:0> |         |         |         |         |
| bit 7 |     |     |          |         |         |         | bit 0   |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

u = unchanged

bit 7-5

**Unimplemented:** Read as '0'

bit 4-0

**GSS<4:0>:** Timerx Gate Source Selection bits

| GSS         | Timer1                 | Timer3                 | Timer5                 |
|-------------|------------------------|------------------------|------------------------|
|             | Gate Source            | Gate Source            | Gate Source            |
| 11111-11011 | Reserved               | Reserved               | Reserved               |
| 11010       | CLC4_out               | CLC4_out               | CLC4_out               |
| 11001       | CLC3_out               | CLC3_out               | CLC3_out               |
| 11000       | CLC2_out               | CLC2_out               | CLC2_out               |
| 10111       | CLC1_out               | CLC1_out               | CLC1_out               |
| 10110       | ZCDOUT                 | ZCDOUT                 | ZCDOUT                 |
| 10101       | CMP2OUT                | CMP2OUT                | CMP2OUT                |
| 10100       | CMP1OUT                | CMP1OUT                | CMP1OUT                |
| 10011       | NCO1OUT                | NCO1OUT                | NCO1OUT                |
| 10010-10001 | Reserved               | Reserved               | Reserved               |
| 10000       | PWM8OUT                | PWM8OUT                | PWM8OUT                |
| 01111       | PWM7OUT                | PWM7OUT                | PWM7OUT                |
| 01110       | PWM6OUT                | PWM6OUT                | PWM6OUT                |
| 01101       | PWM5OUT                | PWM5OUT                | PWM5OUT                |
| 01100       | CCP4OUT                | CCP4OUT                | CCP4OUT                |
| 01011       | CCP3OUT                | CCP3OUT                | CCP3OUT                |
| 01010       | CCP2OUT                | CCP2OUT                | CCP2OUT                |
| 01001       | CCP1OUT                | CCP1OUT                | CCP1OUT                |
| 01000       | SMT1_match             | SMT1_match             | SMT1_match             |
| 00111       | TMR6OUT (post-scaled)  | TMR6OUT (post-scaled)  | TMR6OUT (post-scaled)  |
| 00110       | TMR5 overflow          | TMR5 overflow          | Reserved               |
| 00101       | TMR4OUT (post-scaled)  | TMR4OUT (post-scaled)  | TMR4OUT (post-scaled)  |
| 00100       | TMR3 overflow          | Reserved               | TMR3 overflow          |
| 00011       | TMR2OUT (post-scaled)  | TMR2OUT (post-scaled)  | TMR2OUT (post-scaled)  |
| 00010       | Reserved               | TMR1 overflow          | TMR1 overflow          |
| 00001       | TMR0 overflow          | TMR0 overflow          | TMR0 overflow          |
| 00000       | Pin selected by T1GPPS | Pin selected by T3GPPS | Pin selected by T5GPPS |

## REGISTER 21-5: TMRxL: TIMERx LOW BYTE REGISTER

|            |         |         |         |         |         |         |         |
|------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-x/x    | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x |
| TMRxL<7:0> |         |         |         |         |         |         |         |
| bit 7      |         |         |         | bit 0   |         |         |         |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-0      **TMRxL<7:0>**:Timerx Low Byte bits

## REGISTER 21-6: TMRxH: TIMERx HIGH BYTE REGISTER

|            |         |         |         |         |         |         |         |
|------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-x/x    | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x |
| TMRxH<7:0> |         |         |         |         |         |         |         |
| bit 7      |         |         |         | bit 0   |         |         |         |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-0      **TMRxH<7:0>**:Timerx High Byte bits

**TABLE 21-3: SUMMARY OF REGISTERS ASSOCIATED WITH TIMER1/3/5 AS A TIMER/COUNTER**

| Name   | Bit 7                                                                      | Bit 6 | Bit 5     | Bit 4    | Bit 3   | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page |
|--------|----------------------------------------------------------------------------|-------|-----------|----------|---------|-------|-------|-------|----------------------|
| TxCON  | —                                                                          | —     | CKPS<1:0> |          | —       | SYNC  | RD16  | ON    | 299                  |
| TxGCON | GE                                                                         | GPOL  | GTM       | GSPM     | GO/DONE | GVAL  | —     | —     | 300                  |
| TxCLK  | —                                                                          | —     | —         | CS<4:0>  |         |       |       |       | 301                  |
| TxGATE | —                                                                          | —     | —         | GSS<4:0> |         |       |       |       | 302                  |
| TMRxL  | Least Significant Byte of the 16-bit TMR3 Register                         |       |           |          |         |       |       |       | 303                  |
| TMRxH  | Holding Register for the Most Significant Byte of the 16-bit TMR3 Register |       |           |          |         |       |       |       | 303                  |

**Legend:** — = Unimplemented location, read as '0'. Shaded cells are not used by TIMER1/3/5.



## 22.0 TIMER2/4/6 MODULE

The Timer2/4/6 modules are 8-bit timers that can operate as free-running period counters or in conjunction with external signals that control start, run, freeze, and reset operation in One-Shot and Monostable modes of operation. Sophisticated waveform control such as pulse density modulation are possible by combining the operation of these timers with other internal peripherals such as the comparators and CCP modules. Features of the timer include:

- 8-bit timer register
- 8-bit period register
- Selectable external hardware timer resets
- Programmable prescaler (1:1 to 1:128)
- Programmable postscaler (1:1 to 1:16)
- Selectable synchronous/asynchronous operation
- Alternate clock sources
- Interrupt on period

- Three modes of operation:
  - Free Running Period
  - One-Shot
  - Monostable

See [Figure 22-1](#) for a block diagram of Timer2. See [Figure 22-2](#) for the clock source block diagram.

**Note:** Three identical Timer2 modules are implemented on this device. The timers are named Timer2, Timer4, and Timer6. All references to Timer2 apply as well to Timer4 and Timer6. All references to T2PR apply as well to T4PR and T6PR.

**FIGURE 22-1: TIMER2 BLOCK DIAGRAM**

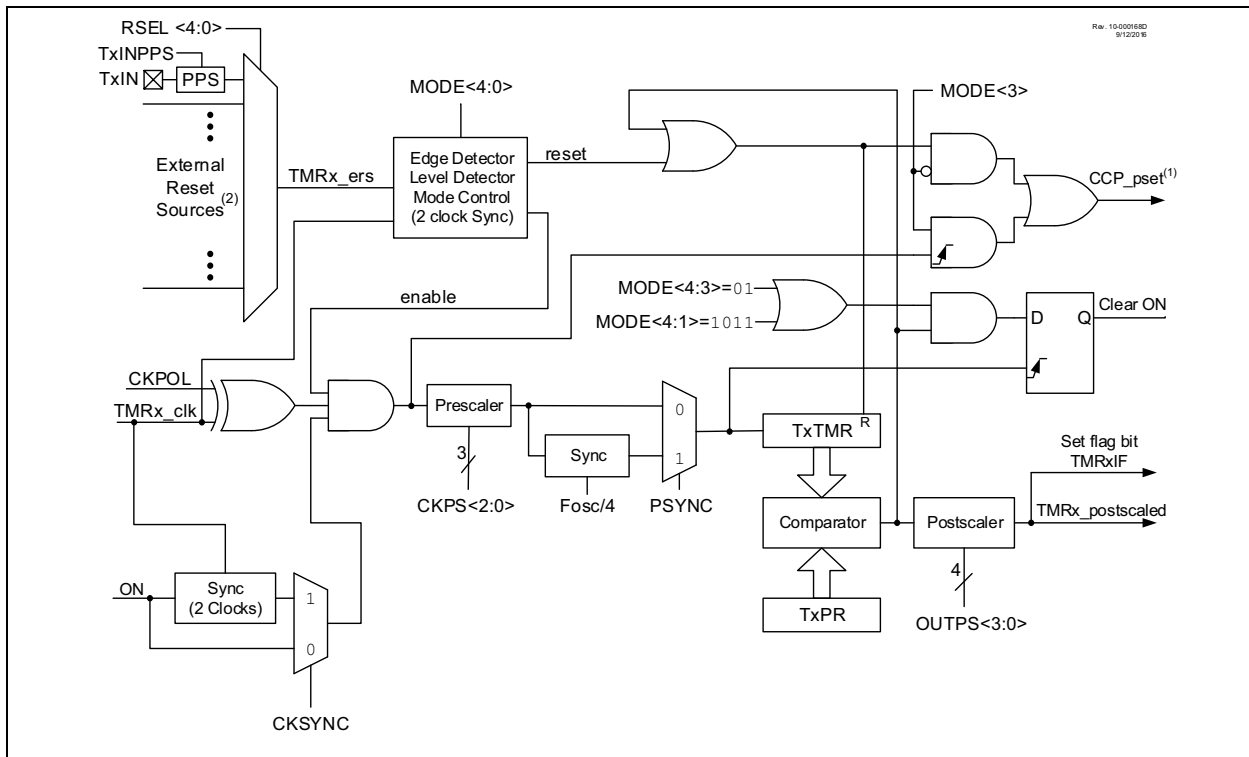
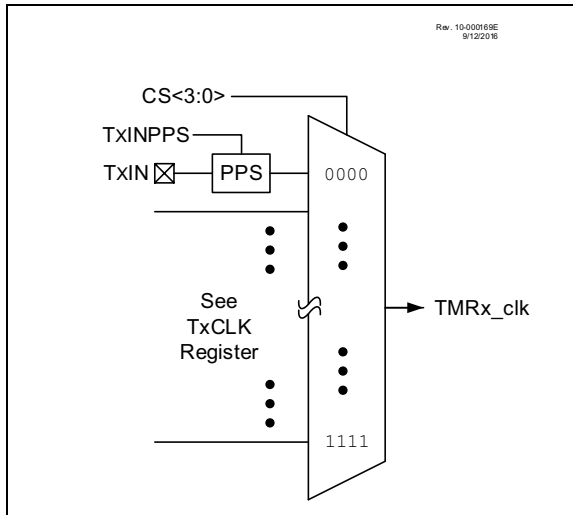


**FIGURE 22-2: TIMER2 CLOCK SOURCE BLOCK DIAGRAM**



## 22.1 Timer2 Operation

Timer2 operates in three major modes:

- Free Running Period
- One-Shot
- Monostable

Within each mode there are several options for starting, stopping, and reset. [Table 22-1](#) lists the options.

In all modes the T2TMR count register is incremented on the rising edge of the clock signal from the programmable prescaler. When T2TMR equals T2PR then a high level is output to the postscaler counter. T2TMR is cleared on the next clock input.

An external signal from hardware can also be configured to gate the timer operation or force a T2TMR count Reset. In gate modes the counter stops when the gate is disabled and resumes when the gate is enabled. In Reset modes the T2TMR count is reset on either the level or edge from the external source.

The T2TMR and T2PR registers are both directly readable and writable. The T2TMR register is cleared and the T2PR register initializes to FFh on any device Reset. Both the prescaler and postscaler counters are cleared on the following events:

- a write to the T2TMR register
- a write to the TxCON register
- any device Reset
- External Reset Source event that resets the timer.

**Note:** T2TMR is not cleared when TxCON is written.

### 22.1.1 FREE RUNNING PERIOD MODE

The value of T2TMR is compared to that of the Period register, T2PR, on each clock cycle. When the two values match, the comparator resets the value of T2TMR to 00h on the next cycle and increments the

output postscaler counter. When the postscaler count equals the value in the OUTPS bits of the TxCON register then a one clock period wide pulse occurs on the T2TMR\_postscaled output, and the postscaler count is cleared.

### 22.1.2 ONE-SHOT MODE

The One-Shot mode is identical to the Free Running Period mode except that the ON bit is cleared and the timer is stopped when T2TMR matches T2PR and will not restart until the T2ON bit is cycled off and on. Postscaler OUTPS values other than 0 are meaningless in this mode because the timer is stopped at the first period event and the postscaler is reset when the timer is restarted.

### 22.1.3 MONOSTABLE MODE

Monostable modes are similar to One-Shot modes except that the ON bit is not cleared and the timer can be restarted by an external Reset event.

## 22.2 Timer2 Output

The Timer2 module's primary output is T2TMR\_postscaled, which pulses for a single T2TMR\_clk period when the postscaler counter matches the value in the OUTPS bits of the TxCON register. The T2PR postscaler is incremented each time the T2TMR value matches the T2PR value. this signal can be selected as an input to several other input modules.

Timer2 is also used by the CCP module for pulse generation in PWM mode. Both the actual T2TMR value as well as other internal signals are sent to the CCP module to properly clock both the period and pulse width of the PWM signal. See [Section 23.0 "Capture/Compare/PWM Module"](#) for more details on setting up Timer2 for use with the CCP, as well as the timing diagrams in [Section 22.5 "Operation Examples"](#) for examples of how the varying Timer2 modes affect CCP PWM output.

## 22.3 External Reset Sources

In addition to the clock source, the Timer2 also takes in an external Reset source. This external Reset source is selected for Timer2, Timer4, and Timer6 with the T2RST, T4RST, and T6RST registers, respectively. This source can control starting and stopping of the timer, as well as resetting the timer, depending on which mode the timer is in. The mode of the timer is controlled by the MODE bits of the T2HLT register. Edge Triggered modes require six Timer clock periods between external triggers. Level Triggered modes require the triggering level to be at least three Timer clock periods long. External triggers are ignored while in Debug Freeze mode.

**TABLE 22-1: TIMER2 OPERATING MODES**

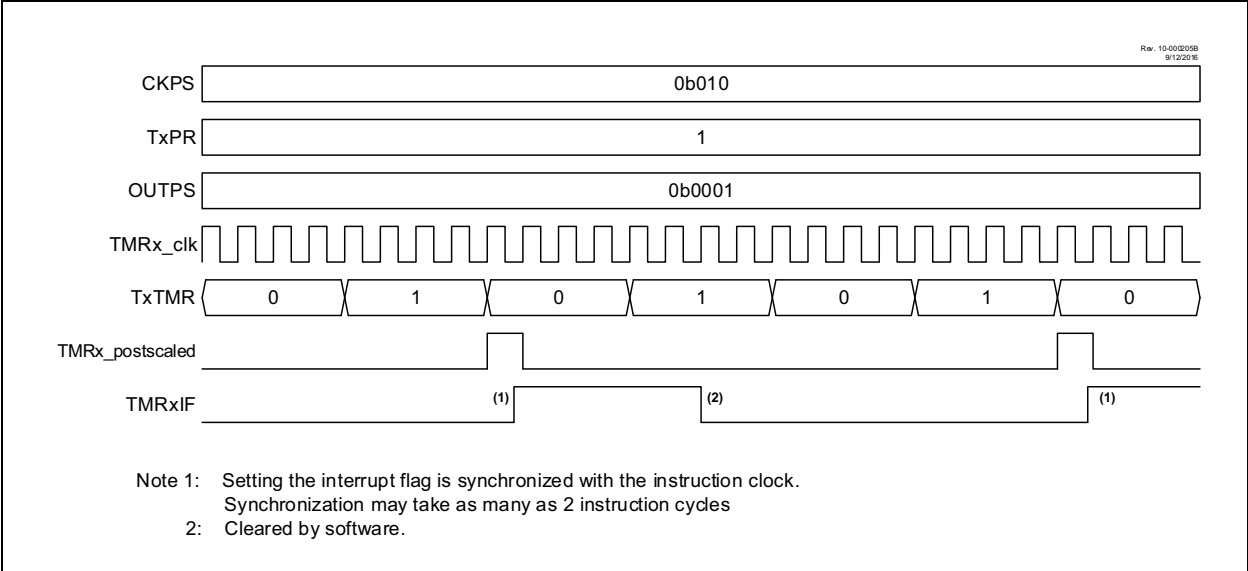
| Mode                | MODE<4:0> |          | Output Operation                                 | Operation                                            | Timer Control                                     |                       |                                                   |                                     |
|---------------------|-----------|----------|--------------------------------------------------|------------------------------------------------------|---------------------------------------------------|-----------------------|---------------------------------------------------|-------------------------------------|
|                     | <4:3>     | <2:0>    |                                                  |                                                      | Start                                             | Reset                 | Stop                                              |                                     |
| Free Running Period | 00        | 000      | Period Pulse                                     | Software gate (Figure 22-6)                          | ON = 1                                            | —                     | ON = 0                                            |                                     |
|                     |           | 001      |                                                  | Hardware gate, active-high (Figure 22-7)             | ON = 1 & TMRx_ers = 1                             | —                     | ON = 0 or TMRx_ers = 0                            |                                     |
|                     |           | 010      |                                                  | Hardware gate, active-low                            | ON = 1 & TMRx_ers = 0                             | —                     | ON = 0 or TMRx_ers = 1                            |                                     |
|                     |           | 011      | Period Pulse with Hardware Reset                 | Rising or Falling Edge Reset                         | ON = 1                                            | TMRx_ers ↓            | ON = 0                                            |                                     |
|                     |           | 100      |                                                  | Rising Edge Reset (Figure 22-8)                      |                                                   | TMRx_ers ↑            |                                                   |                                     |
|                     |           | 101      |                                                  | Falling Edge Reset                                   |                                                   | TMRx_ers ↓            |                                                   |                                     |
|                     |           | 110      |                                                  | Low Level Reset                                      |                                                   | TMRx_ers = 0          | ON = 0 or TMRx_ers = 0                            |                                     |
|                     |           | 111      |                                                  | High Level Reset (Figure 22-9)                       |                                                   | TMRx_ers = 1          | ON = 0 or TMRx_ers = 1                            |                                     |
| One-shot            | 01        | 000      | One-Shot                                         | Software Start (Figure 22-10)                        | ON = 1                                            | —                     | ON = 0 or<br>Next clock after TMRx = PRx (Note 2) |                                     |
|                     |           | 001      | Edge Triggered Start (Note 1)                    | Rising Edge Start (Figure 22-9)                      | ON = 1 & TMRx_ers ↑                               | —                     |                                                   |                                     |
|                     |           | 010      |                                                  | Falling Edge Start                                   | ON = 1 & TMRx_ers ↓                               | —                     |                                                   |                                     |
|                     |           | 011      |                                                  | Any Edge Start                                       | ON = 1 & TMRx_ers ↑↓                              | —                     |                                                   |                                     |
|                     |           | 100      | Edge Triggered Start and Hardware Reset (Note 1) | Rising Edge Start & Rising Edge Reset (Figure 22-12) | ON = 1 & TMRx_ers ↑                               | TMRx_ers ↑            |                                                   |                                     |
|                     |           | 101      |                                                  | Falling Edge Start & Falling Edge Reset              | ON = 1 & TMRx_ers ↓                               | TMRx_ers ↓            |                                                   |                                     |
|                     |           | 110      |                                                  | Rising Edge Start & Low Level Reset (Figure 22-13)   | ON = 1 & TMRx_ers ↑                               | TMRx_ers = 0          |                                                   |                                     |
|                     |           | 111      |                                                  | Falling Edge Start & High Level Reset                | ON = 1 & TMRx_ers ↓                               | TMRx_ers = 1          |                                                   |                                     |
| Mono-stable         | 10        | 000      | Reserved                                         |                                                      |                                                   |                       |                                                   |                                     |
|                     |           | 001      | Edge Triggered Start (Note 1)                    | Rising Edge Start (Figure 22-12)                     | ON = 1 & TMRx_ers ↑                               | —                     | ON=0 or<br>Next clock after TxTMR = TxPR (Note 3) |                                     |
|                     |           | 010      |                                                  | Falling Edge Start                                   | ON = 1 & TMRx_ers ↓                               | —                     |                                                   |                                     |
|                     |           | 011      |                                                  | Any Edge Start                                       | ON = 1 & TMRx_ers ↑↓                              | —                     |                                                   |                                     |
|                     |           | Reserved | 100                                              | Reserved                                             |                                                   |                       |                                                   |                                     |
|                     |           | Reserved | 101                                              | Reserved                                             |                                                   |                       |                                                   |                                     |
|                     |           | One-shot | 110                                              | Level Triggered Start and Hardware Reset             | High Level Start & Low Level Reset (Figure 22-13) | ON = 1 & TMRx_ers = 1 | TMRx_ers = 0                                      | ON = 0 or<br>Held in Reset (Note 2) |
|                     |           |          | 111                                              |                                                      | Low Level Start & High Level Reset                | ON = 1 & TMRx_ers = 0 | TMRx_ers = 1                                      |                                     |
| Reserved            | 11        | xxx      | Reserved                                         |                                                      |                                                   |                       |                                                   |                                     |

- Note 1:** If ON = 0 then an edge is required to restart the timer after ON = 1.  
**Note 2:** When TxTMR = TxPR then the next clock clears ON and stops TxTMR at 00h.  
**Note 3:** When TxTMR = TxPR then the next clock stops TxTMR at 00h but does not clear ON.

22.4 Timer2 Interrupt

Timer2 can also generate a device interrupt. The interrupt is generated when the postscaler counter matches one of 16 postscale options (from 1:1 through 1:16), which is selected with the postscaler control bits, OUTPS of the T2CON register. The interrupt is enabled by setting the T2TMR Interrupt Enable bit, TMR2IE, of the respective PIE register. The interrupt timing is illustrated in Figure 22-3.

FIGURE 22-3: TIMER2 PRESCALER, POSTSCALER, AND INTERRUPT TIMING DIAGRAM



## 22.5 Operation Examples

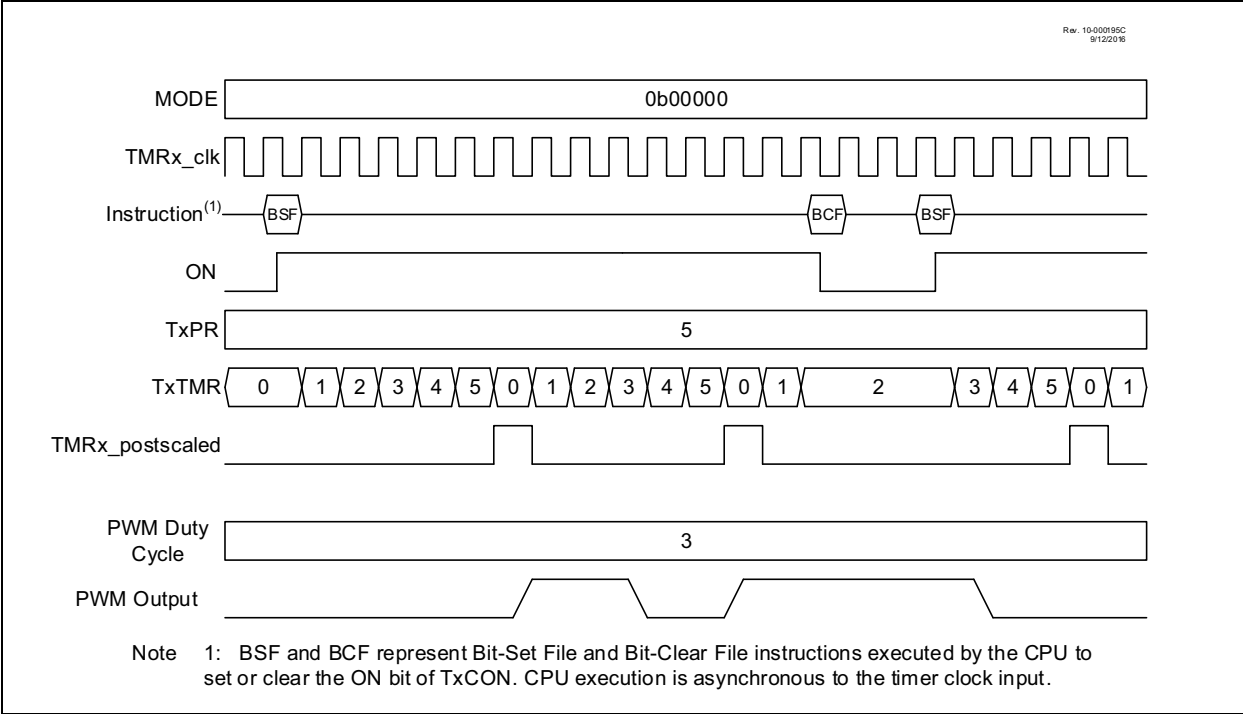
Unless otherwise specified, the following notes apply to the following timing diagrams:

- Both the prescaler and postscaler are set to 1:1 (both the CKPS and OUTPS bits in the T2CON register are cleared).
- The diagrams illustrate any clock except Fosc/4 and show clock-sync delays of at least two full cycles for both ON and T2TMR\_ers. When using Fosc/4, the clock-sync delay is at least one instruction period for T2TMR\_ers; ON applies in the next instruction period.
- ON and T2TMR\_ers are somewhat generalized, and clock-sync delays may produce results that are slightly different than illustrated.
- The PWM Duty Cycle and PWM output are illustrated assuming that the timer is used for the PWM function of the CCP module as described in [Section 23.0 “Capture/Compare/PWM Module”](#) and [Section 24.0 “Pulse-Width Modulation \(PWM\)”](#). The signals are not a part of the T2TMR module.

22.5.1 SOFTWARE GATE MODE

The timer increments with each clock input when ON = 1 and does not increment when ON = 0. When the T2TMR count equals the T2PR period count the timer resets on the next clock and continues counting from 0. Operation with the ON bit software controlled is illustrated in Figure 22-4. With T2PR = 5, the counter advances until T2TMR = 5, and goes to zero with the next clock.

FIGURE 22-4: SOFTWARE GATE MODE TIMING DIAGRAM



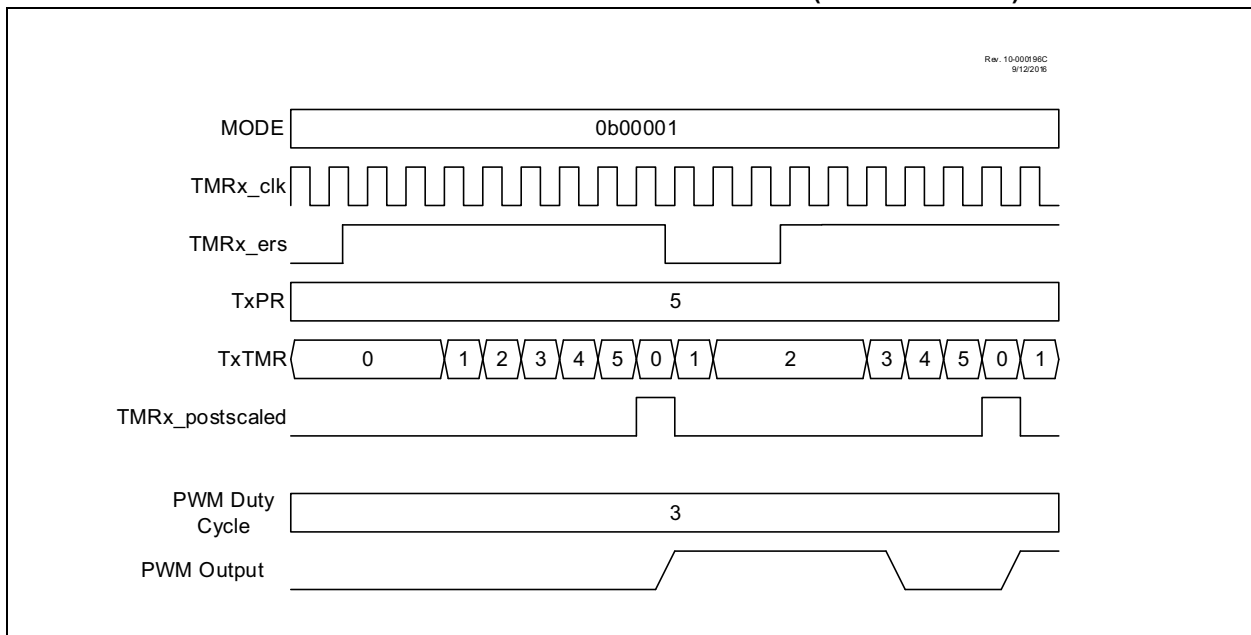
## 22.5.2 HARDWARE GATE MODE

The Hardware Gate modes operate the same as the Software Gate mode except the T2TMR\_ers external signal can also gate the timer. When used with the CCP the gating extends the PWM period. If the timer is stopped when the PWM output is high then the duty cycle is also extended.

When MODE<4:0> = 00001 then the timer is stopped when the external signal is high. When MODE<4:0> = 00010 then the timer is stopped when the external signal is low.

Figure 22-5 illustrates the Hardware Gating mode for MODE<4:0> = 00001 in which a high input level starts the counter.

**FIGURE 22-5: HARDWARE GATE MODE TIMING DIAGRAM (MODE = 00001)**



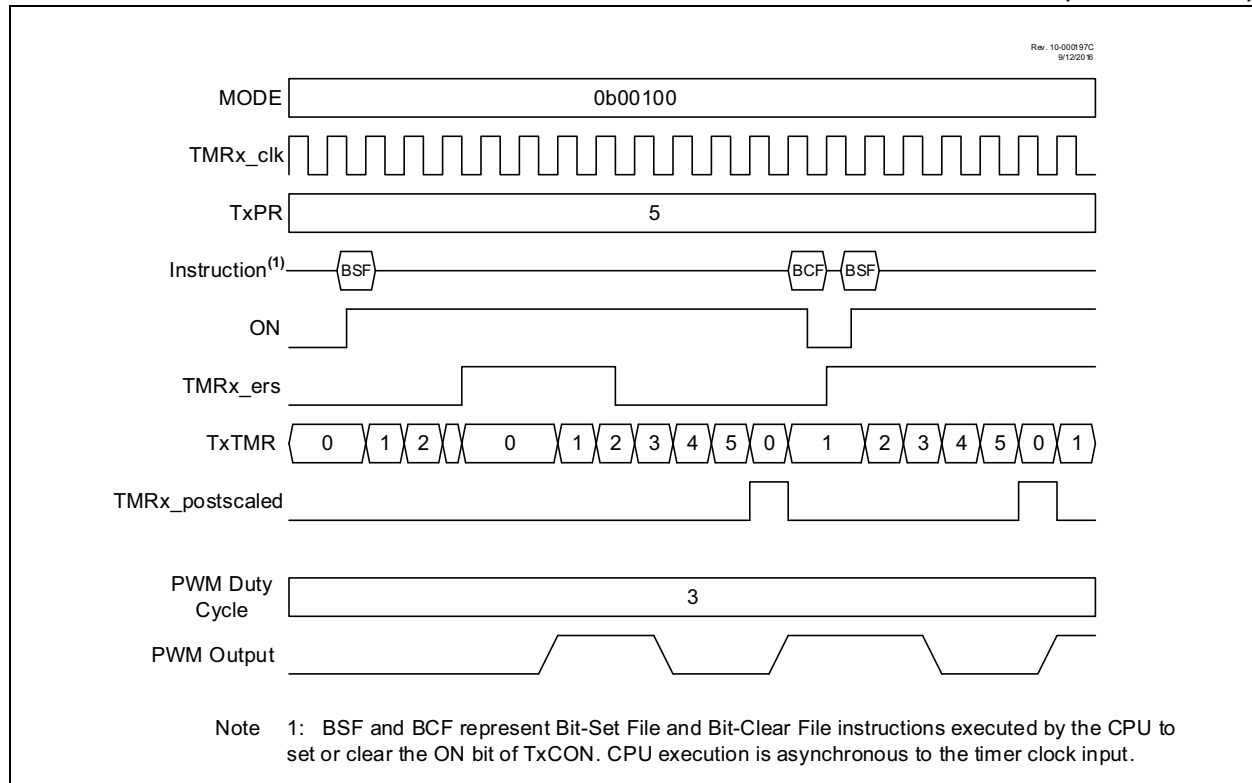
## 22.5.3 EDGE-TRIGGERED HARDWARE LIMIT MODE

In Hardware Limit mode the timer can be reset by the TMRx\_ers external signal before the timer reaches the period count. Three types of Resets are possible:

- Reset on rising or falling edge (MODE<4:0> = 00011)
- Reset on rising edge (MODE<4:0> = 0010)
- Reset on falling edge (MODE<4:0> = 00101)

When the timer is used in conjunction with the CCP in PWM mode then an early Reset shortens the period and restarts the PWM pulse after a two clock delay. Refer to [Figure 22-6](#).

**FIGURE 22-6: EDGE TRIGGERED HARDWARE LIMIT MODE TIMING DIAGRAM (MODE=00100)**





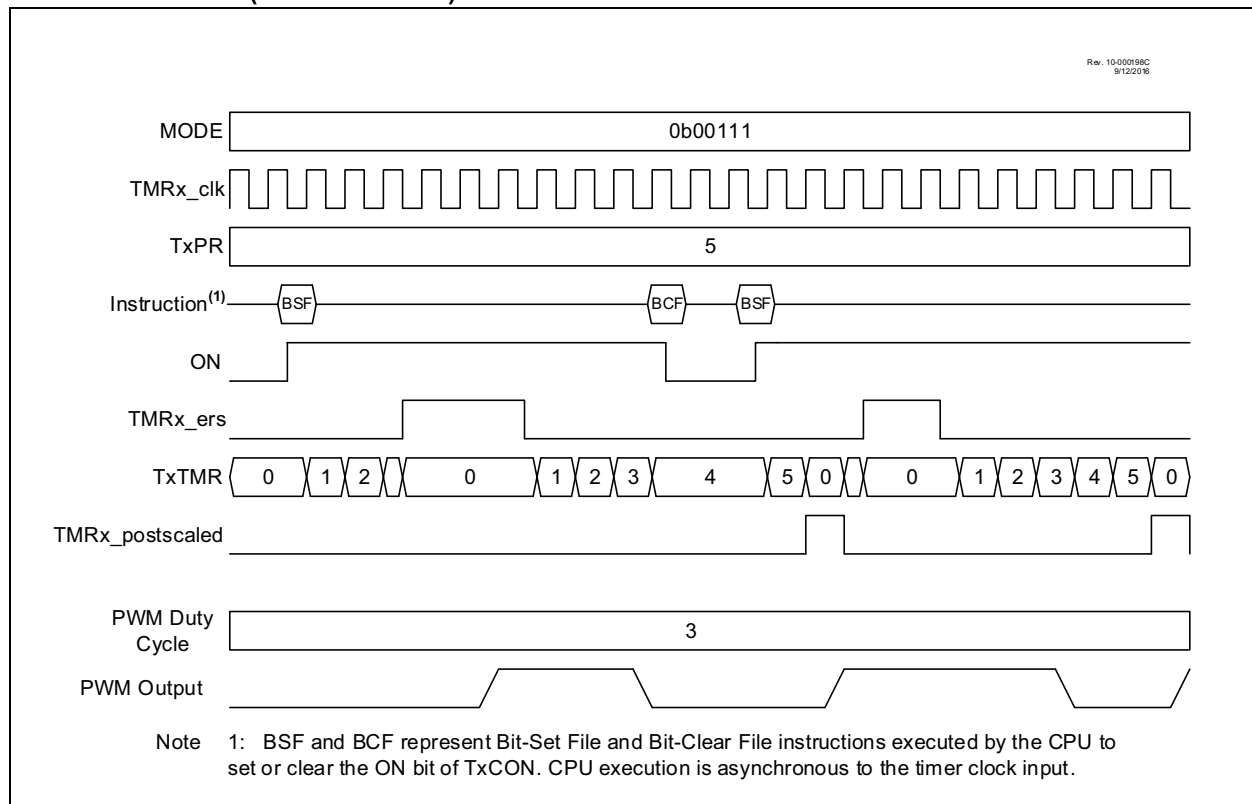
## 22.5.4 LEVEL-TRIGGERED HARDWARE LIMIT MODE

In the level triggered Hardware Limit Timer modes the counter is reset by high or low levels of the external signal TMR2\_ers, as shown in Figure 22-7. Selecting MODE<4:0> = 00110 will cause the timer to reset on a low level external signal. Selecting MODE<4:0> = 00111 will cause the timer to reset on a high level external signal. In the example, the counter is reset while TMR2\_ers = 1. ON is controlled by BSF and BCF instructions. When ON=0 the external signal is ignored.

When the CCP uses the timer as the PWM time base then the PWM output will be set high when the timer starts counting and then set low only when the timer count matches the CCPRx value. The timer is reset when either the timer count matches the T2PR value or two clock periods after the external Reset signal goes true and stays true.

The timer starts counting, and the PWM output is set high, on either the clock following the T2PR match or two clocks after the external Reset signal relinquishes the Reset. The PWM output will remain high until the timer counts up to match the CCPRx pulse width value. If the external Reset signal goes true while the PWM output is high then the PWM output will remain high until the Reset signal is released allowing the timer to count up to match the CCPRx value.

**FIGURE 22-7: LEVEL TRIGGERED HARDWARE LIMIT MODE TIMING DIAGRAM (MODE = 00111)**

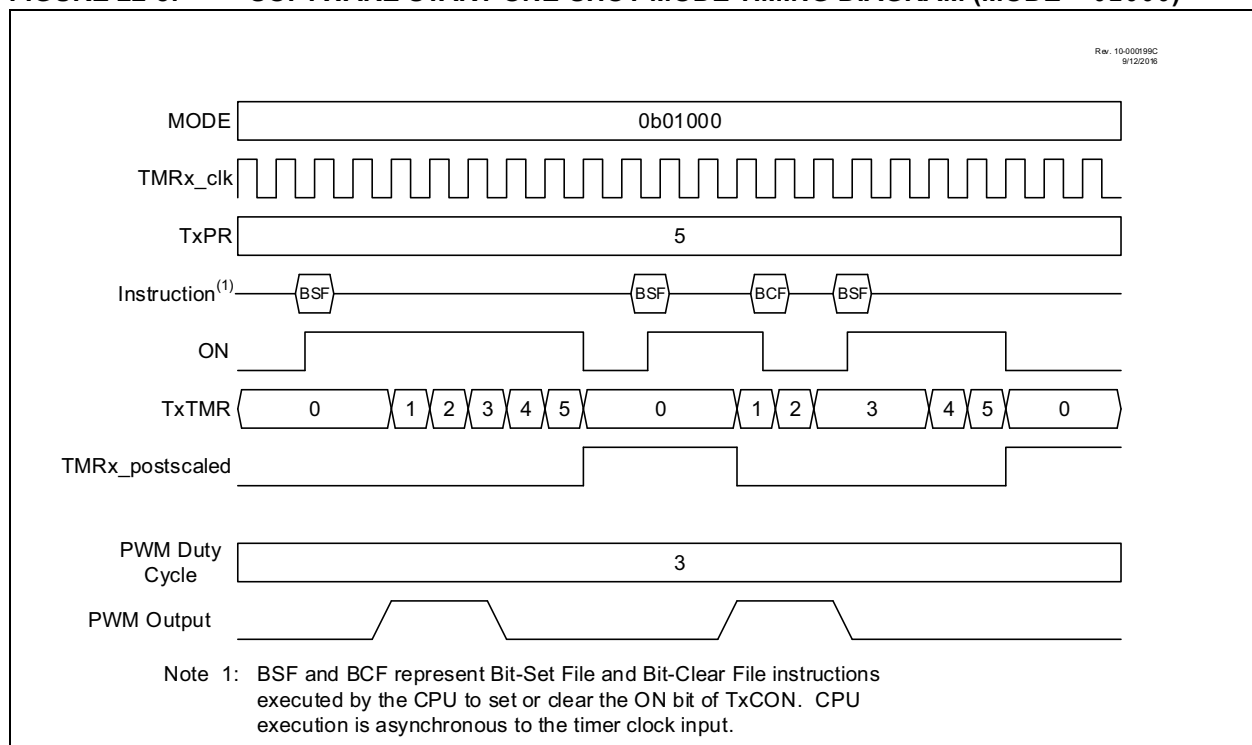


### 22.5.5 SOFTWARE START ONE-SHOT MODE

In One-Shot mode the timer resets and the ON bit is cleared when the timer value matches the T2PR period value. The ON bit must be set by software to start another timer cycle. Setting `MODE<4:0> = 01000` selects One-Shot mode which is illustrated in [Figure 22-8](#). In the example, ON is controlled by BSF and BCF instructions. In the first case, a BSF instruction sets ON and the counter runs to completion and clears ON. In the second case, a BSF instruction starts the cycle, BCF/BSF instructions turn the counter off and on during the cycle, and then it runs to completion.

When One-Shot mode is used in conjunction with the CCP PWM operation the PWM pulse drive starts concurrent with setting the ON bit. Clearing the ON bit while the PWM drive is active will extend the PWM drive. The PWM drive will terminate when the timer value matches the CCPRx pulse width value. The PWM drive will remain off until software sets the ON bit to start another cycle. If software clears the ON bit after the CCPRx match but before the T2PR match then the PWM drive will be extended by the length of time the ON bit remains cleared. Another timing cycle can only be initiated by setting the ON bit after it has been cleared by a T2PR period count match.

**FIGURE 22-8: SOFTWARE START ONE-SHOT MODE TIMING DIAGRAM (MODE = 01000)**



## 22.5.6 EDGE-TRIGGERED ONE-SHOT MODE

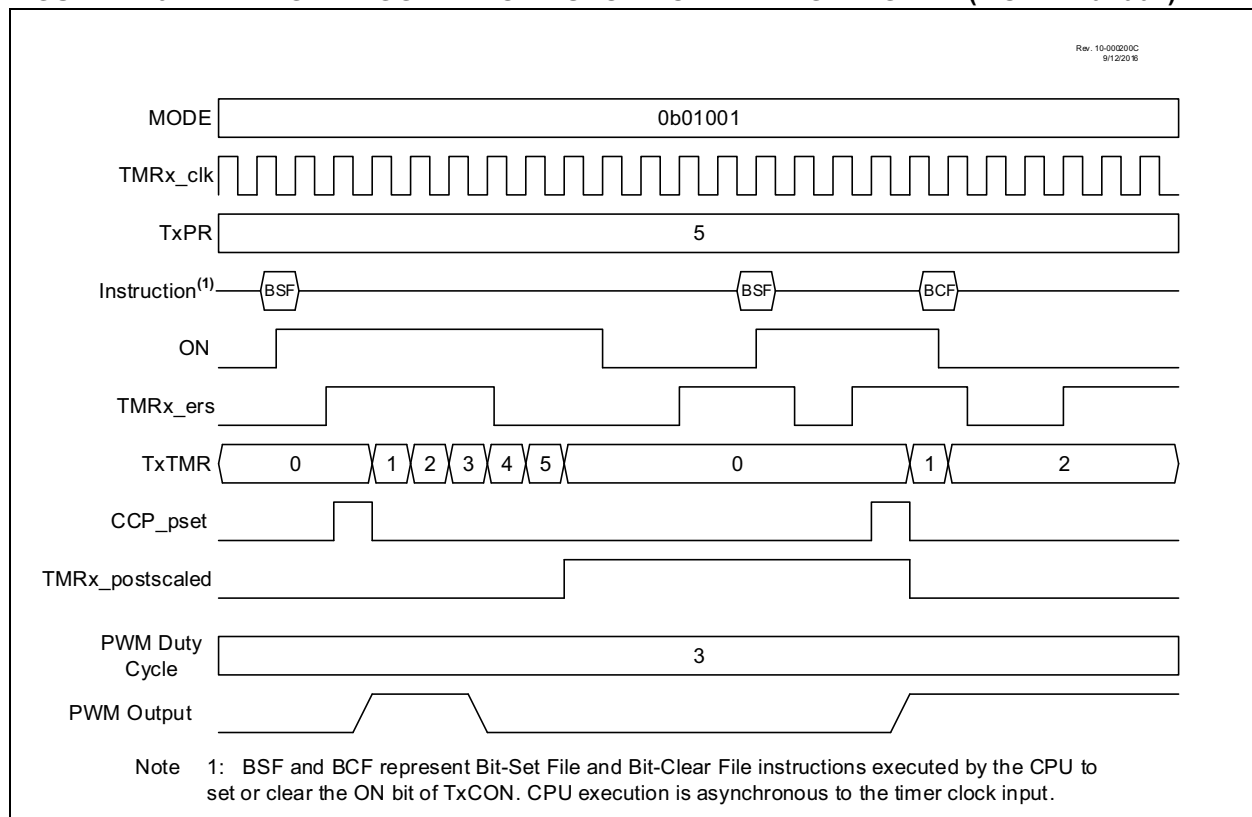
The Edge-Triggered One-Shot modes start the timer on an edge from the external signal input, after the ON bit is set, and clear the ON bit when the timer matches the T2PR period value. The following edges will start the timer:

- Rising edge (MODE<4:0> = 01001)
- Falling edge (MODE<4:0> = 01010)
- Rising or Falling edge (MODE<4:0> = '01011')

If the timer is halted by clearing the ON bit then another TMRx\_ers edge is required after the ON bit is set to resume counting. [Figure 22-9](#) illustrates operation in the rising edge One-Shot mode.

When Edge-Triggered One-Shot mode is used in conjunction with the CCP then the edge-trigger will activate the PWM drive and the PWM drive will deactivate when the timer matches the CCPRx pulse width value and stay deactivated when the timer halts at the T2PR period count match.

**FIGURE 22-9: EDGE TRIGGERED ONE-SHOT MODE TIMING DIAGRAM (MODE = 01001)**



### 22.5.7 EDGE-TRIGGERED HARDWARE LIMIT ONE-SHOT MODE

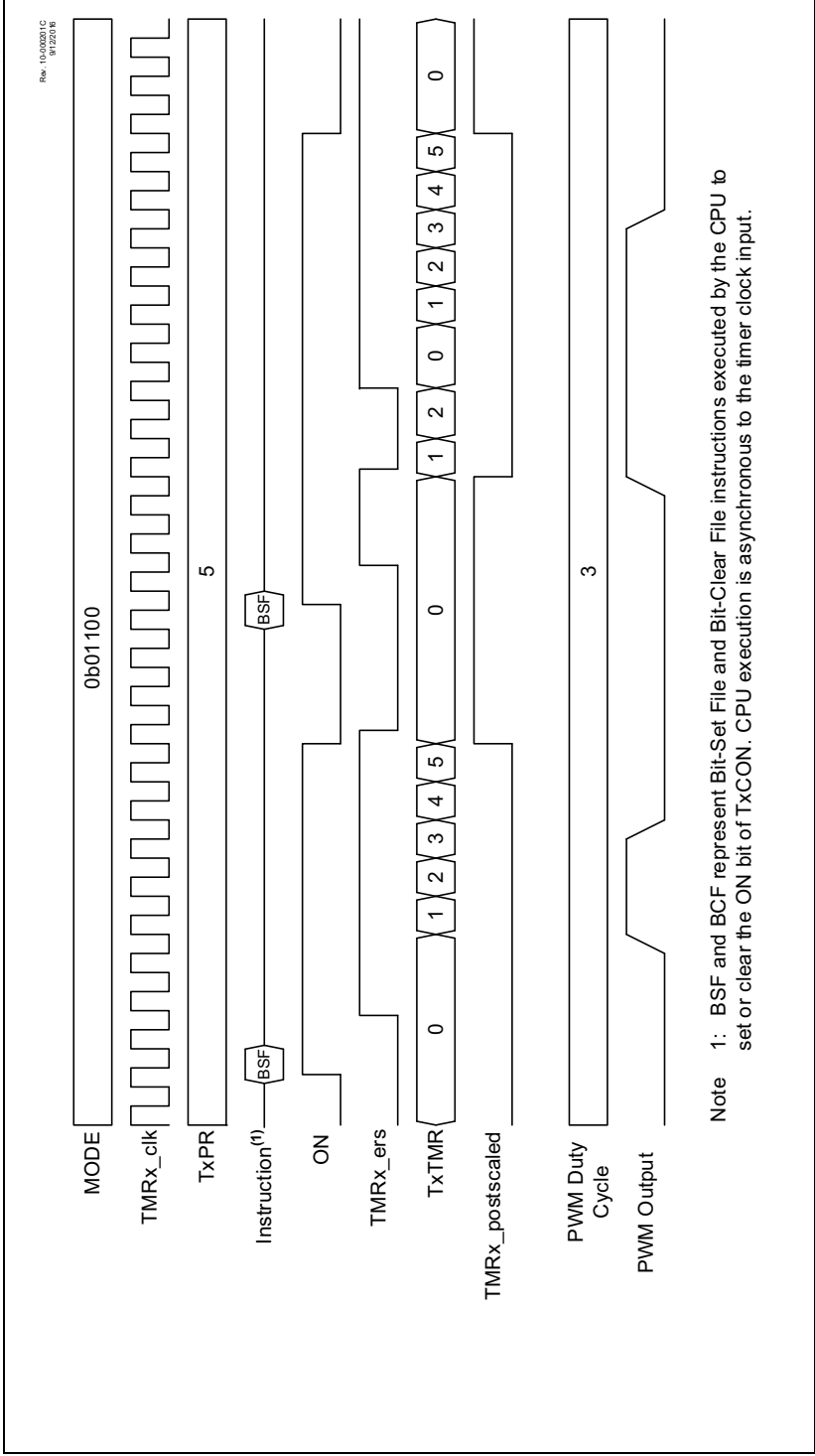
In Edge-Triggered Hardware Limit One-Shot modes the timer starts on the first external signal edge after the ON bit is set and resets on all subsequent edges. Only the first edge after the ON bit is set is needed to start the timer. The counter will resume counting automatically two clocks after all subsequent external Reset edges. Edge triggers are as follows:

- Rising edge Start and Reset (MODE<4:0> = 01100)
- Falling edge Start and Reset (MODE<4:0> = 01101)

The timer resets and clears the ON bit when the timer period value. External signal edges will have no effect on the ON bit. [Figure 22-10](#) illustrates the rising edge operation.

When this mode is used in conjunction with the CCP trigger, and all subsequent Reset edges, will activate the drive will deactivate when the timer matches the CCP period. The timer will stay deactivated until the timer halts at the T2PR period. The timer will reset the timer before the match occurs.

**FIGURE 22-10: EDGE TRIGGERED HARDWARE LIMIT ONE-SHOT MODE TIMING DIAGRAM (MODE = 01100))**



### 22.5.8 LEVEL RESET, EDGE-TRIGGERED HARDWARE LIMIT ONE-SHOT MODES

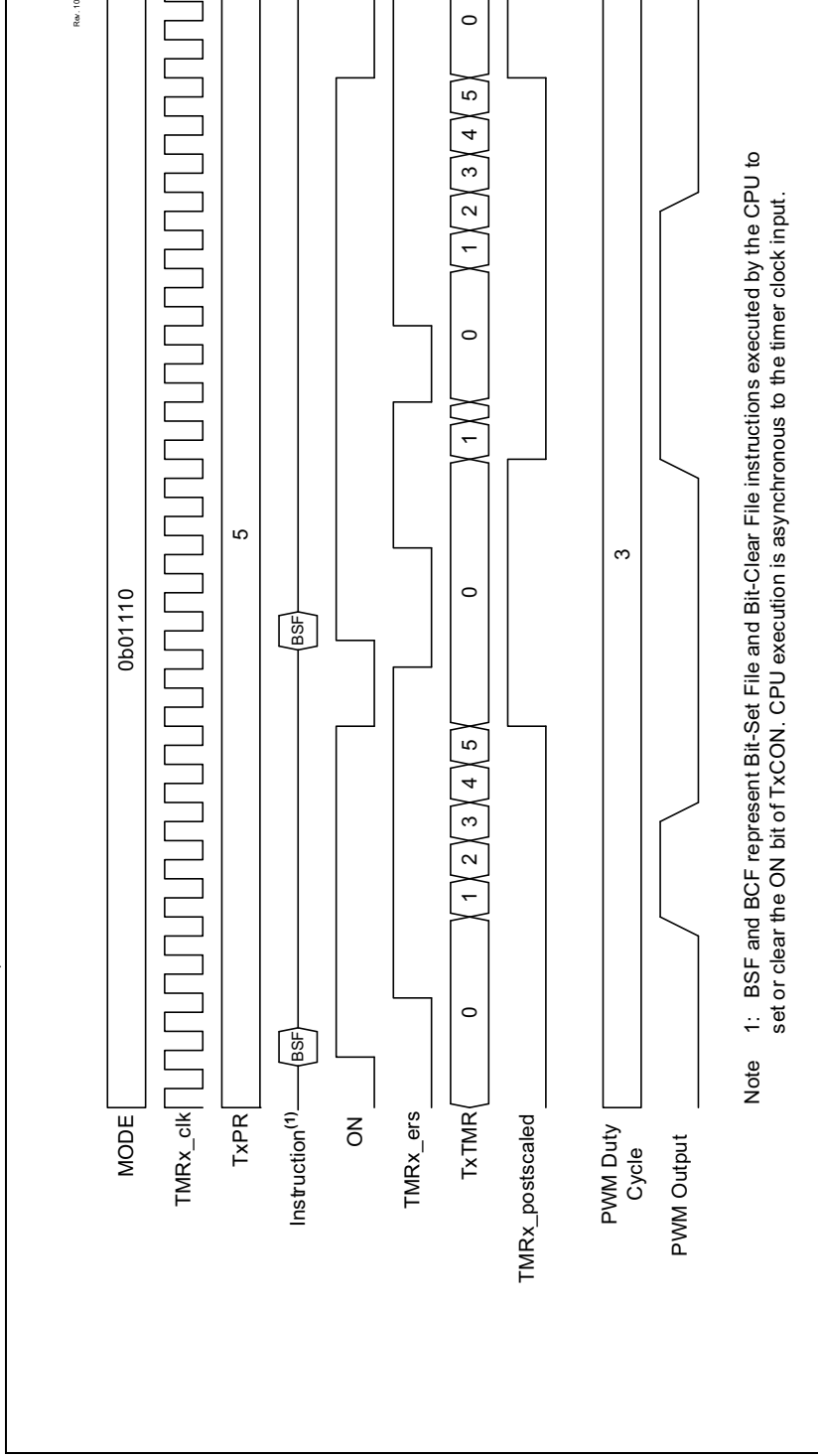
In Level Triggered One-Shot mode the timer count is reset on the external signal level and starts counting on the rising/falling edge of the transition from reset level to the active level while the ON bit is set. Reset levels are selected as follows:

- Low reset level (MODE<4:0> = 01110)
- High reset level (MODE<4:0> = 01111)

When the timer count matches the T2PR period count the ON bit is cleared. When the ON bit is cleared by software control a new external signal edge is required to start the counter.

When Level Triggered Reset One-Shot mode is used in CCP PWM operation the PWM drive goes active when that starts the timer. The PWM drive goes inactive when the CCPRx pulse-width count. The PWM drive does not clear the timer count at the T2PR period count match.

**FIGURE 22-11: LOW LEVEL RESET, EDGE-TRIGGERED HARDWARE LIMIT ONE-SHOT MODE TIMING DIAGRAM**



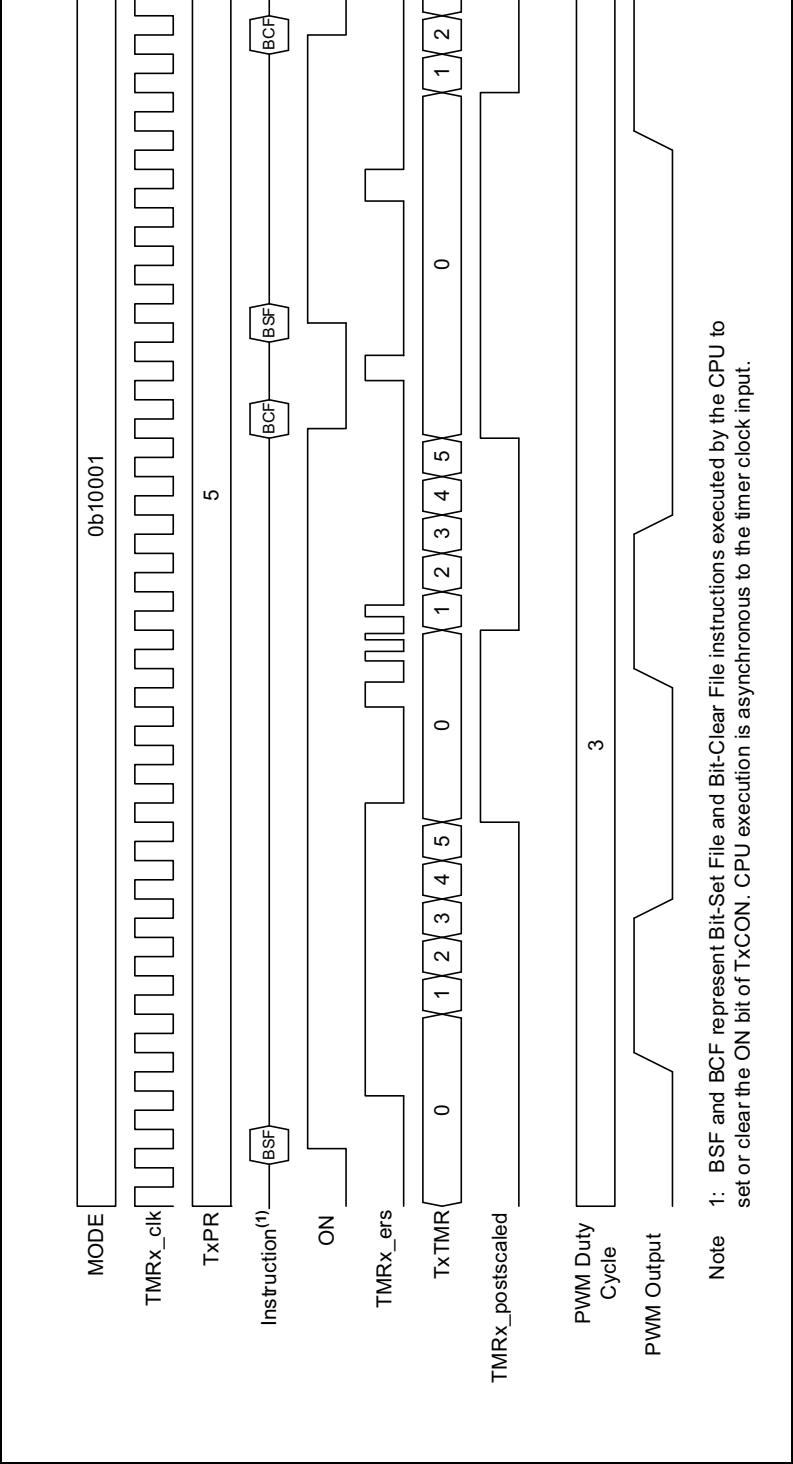
### 22.5.9 EDGE-TRIGGERED MONOSTABLE MODES

The Edge-Triggered Monostable modes start the timer on an edge from the external Reset signal input, after the ON bit is set, and stop incrementing the timer when the timer matches the T2PR period value. The following edges will start the timer:

- Rising edge ( $\text{MODE}\langle 4:0 \rangle = 10001$ )
- Falling edge ( $\text{MODE}\langle 4:0 \rangle = 10010$ )
- Rising or Falling edge ( $\text{MODE}\langle 4:0 \rangle = 10011$ )

When an Edge-Triggered Monostable mode is used, the CCP PWM operation the PWM drive goes active when the edge that starts the timer, but will not go active when the timer matches the T2PR value. While the timer is incrementing, additional Reset signal will not affect the CCP PWM.

**FIGURE 22-12: RISING EDGE-TRIGGERED MONOSTABLE MODE TIMING DIAGRAM (MODE = 10001)**



### 22.5.10 LEVEL-TRIGGERED HARDWARE LIMIT ONE-SHOT MODES

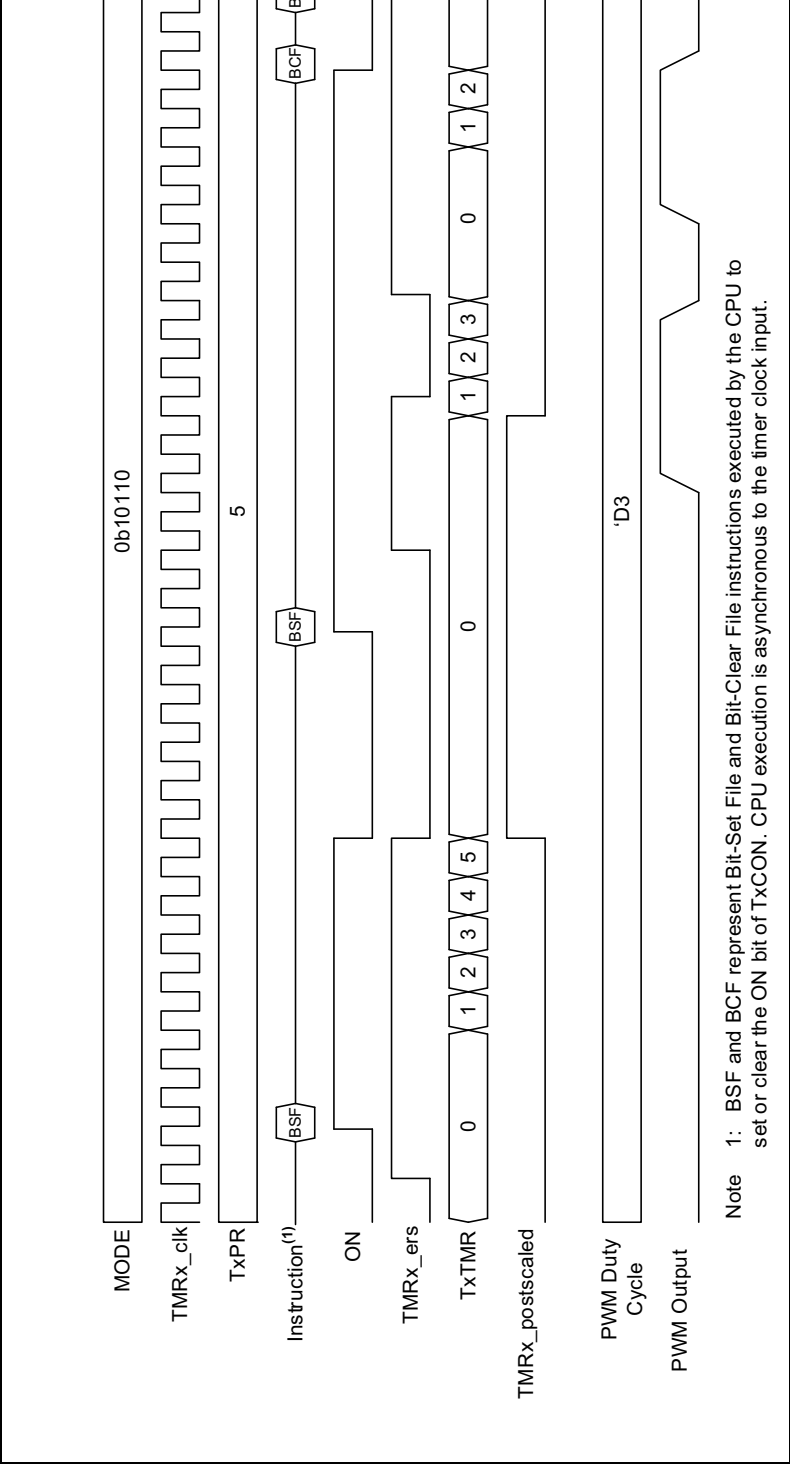
The Level Triggered Hardware Limit One-Shot modes hold the timer in Reset on an external Reset level and start counting when both the ON bit is set and the external signal is not at the Reset level. If one of either the external signal is not in reset or the ON bit is set then the other signal being set/made active will start the timer. Reset levels are selected as follows:

- Low reset level (MODE<4:0> = 10110)
- High reset level (MODE<4:0> = 10111)

When the timer count matches the T2PR period the ON bit is cleared. When the ON bit is cleared by software control the timer will stay in Reset until both external signal is not at the Reset level.

When Level Triggered Hardware Limit One-Shot mode with the CCP PWM operation the PWM drive goes to high level on the setting of the ON bit, whichever external signal edge or the setting of the ON bit, whichever event occurs first.

**FIGURE 22-13: LEVEL-TRIGGERED HARDWARE LIMIT ONE-SHOT MODE TIMING DIAGRAM (MODE = 10110)**



## 22.6 Timer2 Operation During Sleep

When PSYNC = 1, Timer2 cannot be operated while the processor is in Sleep mode. The contents of the T2TMR and T2PR registers will remain unchanged while processor is in Sleep mode.

When PSYNC = 0, Timer2 will operate in Sleep as long as the clock source selected is also still running. Selecting the LFINTOSC, MFINTOSC, or HFINTOSC oscillator as the timer clock source will keep the selected oscillator running during Sleep.



## 22.7 Register Definitions: Timer2/4/6 Control

Long bit name prefixes for the Timer2/4/6 peripherals are shown in [Table 22-2](#). Refer to [Section 1.3.2.2 “Long Bit Names”](#) for more information.

**TABLE 22-2: OPERATING MODES**

| Peripheral | Bit Name Prefix |
|------------|-----------------|
| Timer2     | T2              |
| Timer4     | T4              |
| Timer6     | T6              |

**REGISTER 22-1: TxCLK: TIMERx CLOCK SELECTION REGISTER**

|       |     |     |     |         |         |         |         |
|-------|-----|-----|-----|---------|---------|---------|---------|
| U-0   | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| —     | —   | —   | —   | CS<3:0> |         |         |         |
| bit 7 |     |     |     | bit 0   |         |         |         |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as ‘0’

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

‘1’ = Bit is set

‘0’ = Bit is cleared

bit 7-4

**Unimplemented:** Read as ‘0’

bit 3-0

**CS<3:0>:** Timerx Clock Selection bits

| CS<3:0> | T2TMR                   | TMR4                    | TMR6                    |
|---------|-------------------------|-------------------------|-------------------------|
|         | Clock Source            | Clock Source            | Clock Source            |
| 1111    | Reserved                | Reserved                | Reserved                |
| 1110    | CLC4_out                | CLC4_out                | CLC4_out                |
| 1101    | CLC3_out                | CLC3_out                | CLC3_out                |
| 1100    | CLC2_out                | CLC2_out                | CLC2_out                |
| 1011    | CLC1_out                | CLC1_out                | CLC1_out                |
| 1010    | ZCD_OUT                 | ZCD_OUT                 | ZCD_OUT                 |
| 1001    | NCO1OUT                 | NCO1OUT                 | NCO1OUT                 |
| 1000    | CLKREF_OUT              | CLKREF_OUT              | CLKREF_OUT              |
| 0111    | SOSC                    | SOSC                    | SOSC                    |
| 0110    | MFINTOSC (32 kHz)       | MFINTOSC (32 kHz)       | MFINTOSC (32 kHz)       |
| 0101    | MFINTOSC (500 kHz)      | MFINTOSC (500 kHz)      | MFINTOSC (500 kHz)      |
| 0100    | LFINTOSC                | LFINTOSC                | LFINTOSC                |
| 0011    | HFINTOSC                | HFINTOSC                | HFINTOSC                |
| 0010    | Fosc                    | Fosc                    | Fosc                    |
| 0001    | Fosc/4                  | Fosc/4                  | Fosc/4                  |
| 0000    | Pin selected by T2INPPS | Pin selected by T4INPPS | Pin selected by T6INPPS |

## REGISTER 22-2: TxRST: TIMER2 EXTERNAL RESET SIGNAL SELECTION REGISTER

|       |     |     |           |         |         |         |         |
|-------|-----|-----|-----------|---------|---------|---------|---------|
| U-0   | U-0 | U-0 | R/W-0/0   | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| —     | —   | —   | RSEL<4:0> |         |         |         |         |
| bit 7 |     |     |           |         |         |         |         |
|       |     |     |           |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-5

**Unimplemented:** Read as '0'

bit 4-0

**RSEL<4:0>:** Timer2 External Reset Signal Source Selection bits

| RSEL<4:0>   | T2TMR                   | TMR4                    | TMR6                    |
|-------------|-------------------------|-------------------------|-------------------------|
|             | Reset Source            | Reset Source            | Reset Source            |
| 11111-11001 | Reserved                | Reserved                | Reserved                |
| 11000       | UART2_tx_edge           | UART2_tx_edge           | UART2_tx_edge           |
| 10111       | UART2_rx_edge           | UART2_rx_edge           | UART2_rx_edge           |
| 10110       | UART1_tx_edge           | UART1_tx_edge           | UART1_tx_edge           |
| 10101       | UART1_rx_edge           | UART1_rx_edge           | UART1_rx_edge           |
| 10100       | CLC4_out                | CLC4_out                | CLC4_out                |
| 10011       | CLC3_out                | CLC3_out                | CLC3_out                |
| 10010       | CLC2_out                | CLC2_out                | CLC2_out                |
| 10001       | CLC1_out                | CLC1_out                | CLC1_out                |
| 10000       | ZCD_OUT                 | ZCD_OUT                 | ZCD_OUT                 |
| 01111       | CMP2OUT                 | CMP2OUT                 | CMP2OUT                 |
| 01110       | CMP1OUT                 | CMP1OUT                 | CMP1OUT                 |
| 01101-01100 | Reserved                | Reserved                | Reserved                |
| 01011       | PWM8OUT                 | PWM8OUT                 | PWM8OUT                 |
| 01010       | PWM7OUT                 | PWM7OUT                 | PWM7OUT                 |
| 01001       | PWM6OUT                 | PWM6OUT                 | PWM6OUT                 |
| 01000       | PWM5OUT                 | PWM5OUT                 | PWM5OUT                 |
| 00111       | CCP4OUT                 | CCP4OUT                 | CCP4OUT                 |
| 00110       | CCP3OUT                 | CCP3OUT                 | CCP3OUT                 |
| 00101       | CCP2OUT                 | CCP2OUT                 | CCP2OUT                 |
| 00100       | CCP1OUT                 | CCP1OUT                 | CCP1OUT                 |
| 00011       | TMR6 postscaled         | TMR6 postscaled         | Reserved                |
| 00010       | TMR4 postscaled         | Reserved                | TMR4 postscaled         |
| 00001       | Reserved                | T2TMR postscaled        | T2TMR postscaled        |
| 00000       | Pin selected by T2INPPS | Pin selected by T4INPPS | Pin selected by T6INPPS |

## REGISTER 22-3: TxTMR: TIMERx COUNTER REGISTER

|           |         |         |         |         |         |         |         |
|-----------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0   | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| TMRx<7:0> |         |         |         |         |         |         |         |
| bit 7     |         |         |         | bit 0   |         |         |         |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0      **TMRx<7:0>**: Timerx Counter bits

## REGISTER 22-4: TxPR: TIMERx PERIOD REGISTER

|          |         |         |         |         |         |         |         |
|----------|---------|---------|---------|---------|---------|---------|---------|
| R/W-1/1  | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
| PRx<7:0> |         |         |         |         |         |         |         |
| bit 7    |         |         |         | bit 0   |         |         |         |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0      **PRx<7:0>**: Timerx Period Register bits

## REGISTER 22-5: TxCON: TIMERx CONTROL REGISTER

| R/W/HC-0/0 | R/W-0/0   | R/W-0/0 | R/W-0/0 | R/W-0/0    | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|------------|-----------|---------|---------|------------|---------|---------|---------|
| ON         | CKPS<2:0> |         |         | OUTPS<3:0> |         |         |         |
| bit 7      |           |         |         |            |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

HC = Bit is cleared by hardware

bit 7 **ON:** Timerx On bit<sup>(1)</sup>

1 = Timerx is On

0 = Timerx is Off: all counters and state machines are reset

bit 6-4 **CKPS<2:0>:** Timerx-type Clock Prescale Select bits

111 = 1:128 Prescaler

110 = 1:64 Prescaler

101 = 1:32 Prescaler

100 = 1:16 Prescaler

011 = 1:8 Prescaler

010 = 1:4 Prescaler

001 = 1:2 Prescaler

000 = 1:1 Prescaler

bit 3-0 **OUTPS<3:0>:** Timerx Output Postscaler Select bits

1111 = 1:16 Postscaler

1110 = 1:15 Postscaler

1101 = 1:14 Postscaler

1100 = 1:13 Postscaler

1011 = 1:12 Postscaler

1010 = 1:11 Postscaler

1001 = 1:10 Postscaler

1000 = 1:9 Postscaler

0111 = 1:8 Postscaler

0110 = 1:7 Postscaler

0101 = 1:6 Postscaler

0100 = 1:5 Postscaler

0011 = 1:4 Postscaler

0010 = 1:3 Postscaler

0001 = 1:2 Postscaler

0000 = 1:1 Postscaler

**Note 1:** In certain modes, the ON bit will be auto-cleared by hardware. See [Section 22.1.2 "One-Shot Mode"](#).

**REGISTER 22-6: TxHLT: TIMERx HARDWARE LIMIT CONTROL REGISTER**

|         |         |         |           |         |         |         |         |
|---------|---------|---------|-----------|---------|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0   | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| PSYNC   | CKPOL   | CKSYNC  | MODE<4:0> |         |         |         |         |
| bit 7   |         |         |           |         |         |         | bit 0   |

**Legend:**

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

- bit 7      **PSYNC:** Timerx Prescaler Synchronization Enable bit<sup>(1, 2)</sup>  
1 = TxTMR Prescaler Output is synchronized to Fosc/4  
0 = TxTMR Prescaler Output is not synchronized to Fosc/4
- bit 6      **CKPOL:** Timerx Clock Polarity Selection bit<sup>(3)</sup>  
1 = Falling edge of input clock clocks timer/prescaler  
0 = Rising edge of input clock clocks timer/prescaler
- bit 5      **CKSYNC:** Timerx Clock Synchronization Enable bit<sup>(4, 5)</sup>  
1 = ON register bit is synchronized to T2TMR\_clk input  
0 = ON register bit is not synchronized to T2TMR\_clk input
- bit 4-0    **MODE<4:0>:** Timerx Control Mode Selection bits<sup>(6, 7)</sup>  
See [Table 22-1](#) for all operating modes.

- Note 1:** Setting this bit ensures that reading TxTMR will return a valid data value.
- 2:** When this bit is '1', Timer2 cannot operate in Sleep mode.
- 3:** CKPOL should not be changed while ON = 1.
- 4:** Setting this bit ensures glitch-free operation when the ON is enabled or disabled.
- 5:** When this bit is set then the timer operation will be delayed by two TxTMR input clocks after the ON bit is set.
- 6:** Unless otherwise indicated, all modes start upon ON = 1 and stop upon ON = 0 (stops occur without affecting the value of TxTMR).
- 7:** When TxTMR = TxPR, the next clock clears TxTMR, regardless of the operating mode.

**TABLE 22-3: SUMMARY OF REGISTERS ASSOCIATED WITH TIMER2**

| Name  | Bit 7                                         | Bit 6     | Bit 5                     | Bit 4     | Bit 3      | Bit 2   | Bit 1 | Bit 0 | Register on Page |
|-------|-----------------------------------------------|-----------|---------------------------|-----------|------------|---------|-------|-------|------------------|
| TxPR  | Timer2 Module Period Register                 |           |                           |           |            |         |       |       | 306*             |
| TxTMR | Holding Register for the 8-bit T2TMR Register |           |                           |           |            |         |       |       | 306*             |
| TxCON | ON                                            | CKPS<2:0> |                           |           | OUTPS<3:0> |         |       |       | 324              |
| TxCLK | —                                             | —         | —                         | —         | —          | CS<2:0> |       |       | 321              |
| TxRST | —                                             | —         | —                         | —         | RSEL<3:0>  |         |       |       | 322              |
| TxHLT | $\overline{\text{PSYNC}}$                     | CPOL      | $\overline{\text{CSYNC}}$ | MODE<4:0> |            |         |       | 325   |                  |

**Legend:** — = unimplemented location, read as '0'. Shaded cells are not used for Timer2 module.

\* Page provides register information.

## 23.0 CAPTURE/COMPARE/PWM MODULE

The Capture/Compare/PWM module is a peripheral that allows the user to time and control different events, and to generate Pulse-Width Modulation (PWM) signals. In Capture mode, the peripheral allows the timing of the duration of an event. The Compare mode allows the user to trigger an external event when a predetermined amount of time has expired. The PWM mode can generate pulse-width modulated signals of varying frequency and duty cycle.

This family of devices contains four standard Capture/Compare/PWM modules (CCP1, CCP2, CCP3 and CCP4). Each individual CCP module can select the timer source that controls the module. Each module has an independent timer selection which can be accessed using the CxTSEL bits in the CCPTMRS0 register ([Register 23-2](#)). The default timer selection is TMR1 when using Capture/Compare mode and TMR2 when using PWM mode in the CCPx module.

Please note that the Capture/Compare mode operation is described with respect to TMR1 and the PWM mode operation is described with respect to TMR2 in the following sections.

The Capture and Compare functions are identical for all CCP modules.

**Note 1:** In devices with more than one CCP module, it is very important to pay close attention to the register names used. A number placed after the module acronym is used to distinguish between separate modules. For example, the CCP1CON and CCP2CON control the same operational aspects of two completely different CCP modules.

**2:** Throughout this section, generic references to a CCP module in any of its operating modes may be interpreted as being equally applicable to CCPx module. Register names, module signals, I/O pins, and bit names may use the generic designator 'x' to indicate the use of a numeral to distinguish a particular module, when required.

## 23.1 CCP Module Configuration

Each Capture/Compare/PWM module is associated with a control register (CCPxCON), a capture input selection register (CCPxCAP) and a data register (CCPRx). The data register, in turn, is comprised of two 8-bit registers: CCPRxL (low byte) and CCPRxH (high byte).

### 23.1.1 CCP MODULES AND TIMER RESOURCES

The CCP modules utilize Timers 1 through 6 that vary with the selected mode. Various timers are available to the CCP modules in Capture, Compare or PWM modes, as shown in [Table 23-1](#).

**TABLE 23-1: CCP MODE – TIMER RESOURCE**

| CCP Mode | Timer Resource           |
|----------|--------------------------|
| Capture  | Timer1, Timer3 or Timer5 |
| Compare  |                          |
| PWM      | Timer2, Timer4 or Timer6 |

The assignment of a particular timer to a module is determined by the timer to CCP enable bits in the CCPTMRS0 register (see [Register 23-2](#)). All of the modules may be active at once and may share the same timer resource if they are configured to operate in the same mode (Capture/Compare or PWM) at the same time.

### 23.1.2 OPEN-DRAIN OUTPUT OPTION

When operating in Output mode (the Compare or PWM modes), the drivers for the CCPx pins can be optionally configured as open-drain outputs. This feature allows the voltage level on the pin to be pulled to a higher level through an external pull-up resistor and allows the output to communicate with external circuits without the need for additional level shifters.

## 23.2 Capture Mode

Capture mode makes use of the 16-bit Timer1 resource. When an event occurs on the capture source, the 16-bit CCPRxH:CCPRxL register pair captures and stores the 16-bit value of the TMRxH:TMRxL register pair, respectively. An event is defined as one of the following and is configured by the MODE<3:0> bits of the CCPxCON register:

- Every falling edge of CCPx input
- Every rising edge of CCPx input
- Every 4th rising edge of CCPx input
- Every 16th rising edge of CCPx input
- Every edge of CCPx input (rising or falling)

When a capture is made, the Interrupt Request Flag bit CCPxIF of the respective PIR register is set. The interrupt flag must be cleared in software. If another capture occurs before the value in the CCPRxH:CCPRxL register pair is read, the old captured value is overwritten by the new captured value.

**Note:** If an event occurs during a 2-byte read, the high and low-byte data will be from different events. It is recommended while reading the CCPRxH:CCPRxL register pair to either disable the module or read the register pair twice for data integrity.

Figure 23-1 shows a simplified diagram of the capture operation.

### 23.2.1 CAPTURE SOURCES

In Capture mode, the CCPx pin should be configured as an input by setting the associated TRIS control bit.

**Note:** If the CCPx pin is configured as an output, a write to the port can cause a capture condition.

The capture source is selected by configuring the CTS<2:0> bits of the CCPxCAP register. Refer to CCPxCAP register (Register 23-4) for a list of sources that can be selected.

### 23.2.2 TIMER1 MODE RESOURCE

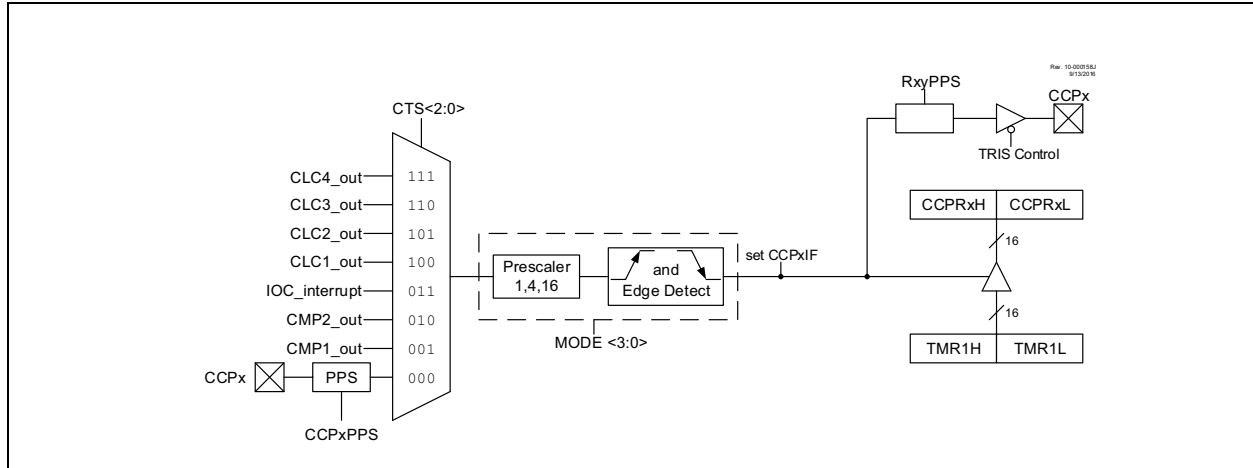
Timer1 must be running in Timer mode or Synchronized Counter mode for the CCP module to use the capture feature. In Asynchronous Counter mode, the capture operation may not work.

- See Section 21.0 “Timer1/3/5 Module with Gate Control” for more information on configuring Timer1.

**Note:** Clocking Timer1 from the system clock (FOSC) should not be used in Capture mode. In order for Capture mode to recognize the trigger event on the CCPx pin, Timer1 must be clocked from the instruction clock (FOSC/4) or from an external clock source.



**FIGURE 23-1: CAPTURE MODE OPERATION BLOCK DIAGRAM**



## 23.2.3 SOFTWARE INTERRUPT MODE

When the Capture mode is changed, a false capture interrupt may be generated. The user should keep the CCPxIE Interrupt Priority bit of the respective PIE register clear to avoid false interrupts. Additionally, the user should clear the CCPxIF interrupt flag bit of the respective PIR register following any change in Operating mode.

## 23.2.4 CAPTURE DURING SLEEP

Capture mode depends upon the Timer1 module for proper operation. There are two options for driving the Timer1 module in Capture mode. It can be driven by the instruction clock ( $F_{osc}/4$ ), or by an external clock source.

When Timer1 is clocked by  $F_{osc}/4$ , Timer1 will not increment during Sleep. When the device wakes from Sleep, Timer1 will continue from its previous state.

Capture mode will operate during Sleep as long as the clock source for Timer1 is active in Sleep.

## 23.3 Compare Mode

Compare mode makes use of the 16-bit Timer1 resource. The 16-bit value of the CCPRxH:CCPRxL register pair is constantly compared against the 16-bit value of the TMRxH:TMRxL register pair. When a match occurs, one of the following events can occur:

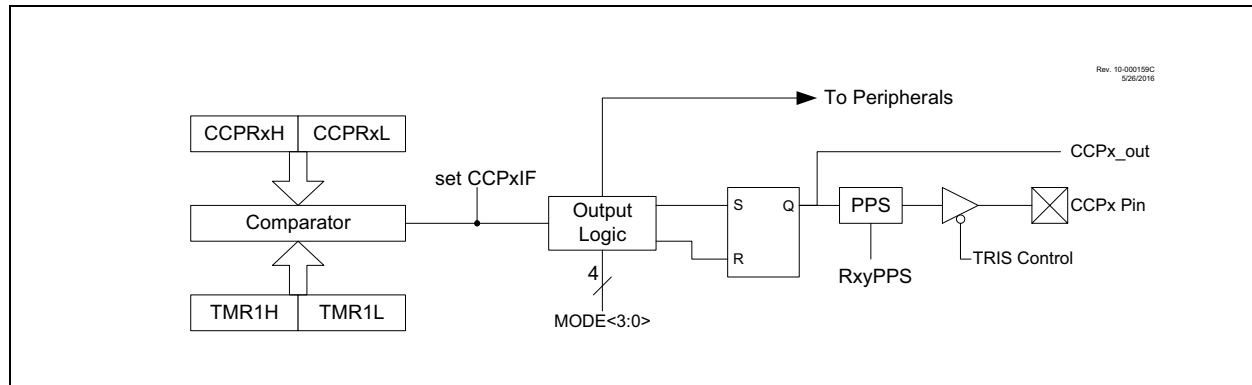
- Toggle the CCPx output, clear TMRx
- Toggle the CCPx output
- Set the CCPx output
- Clear the CCPx output
- Pulse output
- Pulse output, clear TMRx

The action on the pin is based on the value of the MODE<3:0> control bits of the CCPxCON register. At the same time, the interrupt flag CCPxIF bit is set, and an ADC conversion can be triggered, if selected.

All Compare modes can generate an interrupt and trigger an ADC conversion. When MODE = 0b0001 or 0b1011, the CCP resets the TMR register pair.

Figure 23-2 shows a simplified diagram of the compare operation.

**FIGURE 23-2: COMPARE MODE OPERATION BLOCK DIAGRAM**



## 23.3.1 CCPx PIN CONFIGURATION

The software must configure the CCPx pin as an output by clearing the associated TRIS bit and defining the appropriate output pin through the RxyPPS registers. See [Section 17.0 “Peripheral Pin Select \(PPS\) Module”](#) for more details.

**Note:** Clearing the CCPxCON register will force the CCPx compare output latch to the default low level. This is not the PORT I/O data latch.

## 23.3.2 TIMER1 MODE RESOURCE

In Compare mode, Timer1 must be running in either Timer mode or Synchronized Counter mode. The compare operation may not work in Asynchronous Counter mode.

See [Section 21.0 “Timer1/3/5 Module with Gate Control”](#) for more information on configuring Timer1.

**Note:** Clocking Timer1 from the system clock (Fosc) should not be used in Compare mode. In order for Compare mode to recognize the trigger event on the CCPx pin, Timer1 must be clocked from the instruction clock (Fosc/4) or from an external clock source.

## 23.3.3 AUTO-CONVERSION TRIGGER

All CCPx modes set the CCP interrupt flag (CCPxIF). When this flag is set and a match occurs, an auto-conversion trigger can take place if the CCP module is selected as the conversion trigger source.

Refer to [Section 37.2.5 “Auto-Conversion Trigger”](#) for more information.

**Note:** Removing the match condition by changing the contents of the CCPRxH and CCPRxL register pair, between the clock edge that generates the Auto-conversion Trigger and the clock edge that generates the Timer1 Reset, will preclude the Reset from occurring

## 23.3.4 COMPARE DURING SLEEP

Since FOSC is shut down during Sleep mode, the Compare mode will not function properly during Sleep, unless the timer is running. The device will wake on interrupt (if enabled).

## 23.4 PWM Overview

Pulse-Width Modulation (PWM) is a scheme that provides power to a load by switching quickly between fully ON and fully OFF states. The PWM signal resembles a square wave where the high portion of the signal is considered the ON state and the low portion of the signal is considered the OFF state. The high portion, also known as the pulse width, can vary in time and is defined in steps. A larger number of steps applied, which lengthens the pulse width, also supplies more power to the load. Lowering the number of steps applied, which shortens the pulse width, supplies less power. The PWM period is defined as the duration of one complete cycle or the total amount of on and off time combined.

PWM resolution defines the maximum number of steps that can be present in a single PWM period. A higher resolution allows for more precise control of the pulse-width time and in turn the power that is applied to the load.

The term duty cycle describes the proportion of the on time to the off time and is expressed in percentages, where 0% is fully off and 100% is fully on. A lower duty cycle corresponds to less power applied and a higher duty cycle corresponds to more power applied.

[Figure 23-3](#) shows a typical waveform of the PWM signal.

### 23.4.1 STANDARD PWM OPERATION

The standard PWM mode generates a Pulse-Width Modulation (PWM) signal on the CCPx pin with up to ten bits of resolution. The period, duty cycle, and resolution are controlled by the following registers:

- T2PR registers
- T2CON registers
- CCPRxL and CCPRxH registers
- CCPxCON registers

It is required to have Fosc/4 as the clock input to TMR2/4/6 for correct PWM operation. [Figure 23-4](#) shows a simplified block diagram of PWM operation.

**Note:** The corresponding TRIS bit must be cleared to enable the PWM output on the CCPx pin.

**FIGURE 23-3: CCP PWM OUTPUT SIGNAL**

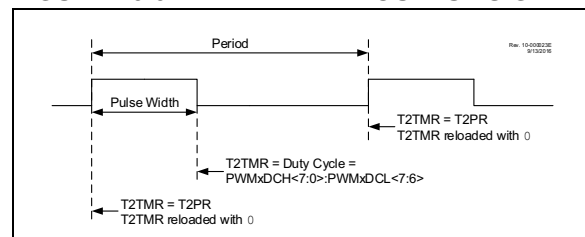
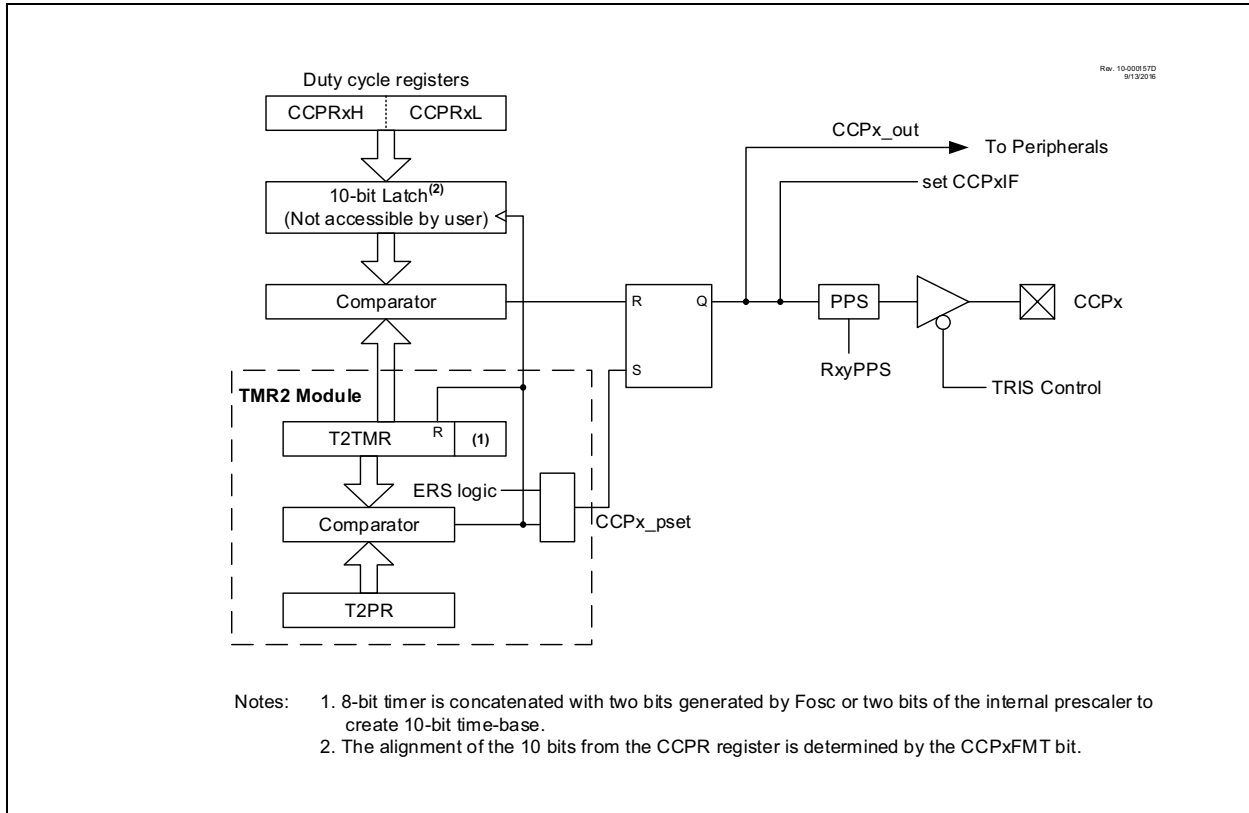


**FIGURE 23-4: SIMPLIFIED PWM BLOCK DIAGRAM**



## 23.4.2 SETUP FOR PWM OPERATION

The following steps should be taken when configuring the CCP module for standard PWM operation:

1. Use the desired output pin RxyPPS control to select CCPx as the source and disable the CCPx pin output driver by setting the associated TRIS bit.
2. Load the T2PR register with the PWM period value.
3. Configure the CCP module for the PWM mode by loading the CCPxCON register with the appropriate values.
4. Load the CCPRxL register, and the CCPRxH register with the PWM duty cycle value and configure the FMT bit of the CCPxCON register to set the proper register alignment.
5. Configure and start Timer2:
  - Clear the TMR2IF interrupt flag bit of the respective PIR register. See Note below.
  - Select the timer clock source to be as Fosc/4 using the T2CLK register. This is required for correct operation of the PWM module.
  - Configure the CKPS bits of the T2CON register with the Timer prescale value.
  - Enable the Timer by setting the ON bit of the T2CON register.
6. Enable PWM output pin:
  - Wait until the Timer overflows and the TMR2IF bit of the PIR4 register is set. See Note below.
  - Enable the CCPx pin output driver by clearing the associated TRIS bit.

**Note:** In order to send a complete duty cycle and period on the first PWM output, the above steps must be included in the setup sequence. If it is not critical to start with a complete PWM signal on the first output, then step 6 may be ignored.

## 23.4.3 TIMER2 TIMER RESOURCE

The PWM standard mode makes use of the 8-bit Timer2 timer resources to specify the PWM period.

## 23.4.4 PWM PERIOD

The PWM period is specified by the T2PR register of Timer2. The PWM period can be calculated using the formula of [Equation 23-1](#).

### EQUATION 23-1: PWM PERIOD

$$\text{PWM Period} = [(T2PR) + 1] \bullet 4 \bullet T_{osc} \bullet (\text{TM R2 Prescale Value})$$

**Note 1:** TOSC = 1/FOSC

When T2TMR is equal to T2PR, the following three events occur on the next increment cycle:

- T2TMR is cleared
- The CCPx pin is set. (Exception: If the PWM duty cycle = 0%, the pin will not be set.)
- The PWM duty cycle is transferred from the CCPRxL/H register pair into a 10-bit buffer.

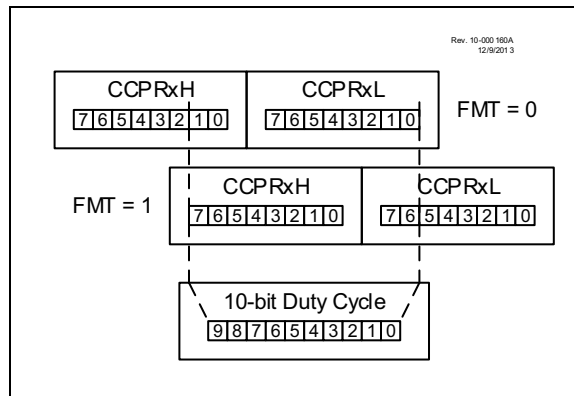
**Note:** The Timer postscaler (see [Section 22.3 “External Reset Sources”](#)) is not used in the determination of the PWM frequency.

## 23.4.5 PWM DUTY CYCLE

The PWM duty cycle is specified by writing a 10-bit value to the CCPRxH:CCPRxL register pair. The alignment of the 10-bit value is determined by the FMT bit of the CCPxCON register (see [Figure 23-5](#)). The CCPRxH:CCPRxL register pair can be written to at any time; however the duty cycle value is not latched into the 10-bit buffer until after a match between T2PR and T2TMR.

[Equation 23-2](#) is used to calculate the PWM pulse width. [Equation 23-3](#) is used to calculate the PWM duty cycle ratio.

**FIGURE 23-5: PWM 10-BIT ALIGNMENT**



**EQUATION 23-2: PULSE WIDTH**

$$\text{Pulse Width} = (\text{CCPRxH:CCPRxL register pair}) \cdot T_{osc} \cdot (\text{TM R2 Prescale Value})$$

**EQUATION 23-3: DUTY CYCLE RATIO**

$$\text{Duty Cycle Ratio} = \frac{(\text{CCPRxH:CCPRxL register pair})}{4(T2PR + 1)}$$

CCPRxH:CCPRxL register pair are used to double buffer the PWM duty cycle. This double buffering provides glitchless PWM operation.

The 8-bit timer T2TMR register is concatenated with either the 2-bit internal system clock (FOSC), or two bits of the prescaler, to create the 10-bit time base. The system clock is used if the Timer2 prescaler is set to 1:1.

When the 10-bit time base matches the CCPRxH:CCPRxL register pair, then the CCPx pin is cleared (see [Figure 23-4](#)).

## 23.4.6 PWM RESOLUTION

The resolution determines the number of available duty cycles for a given period. For example, a 10-bit resolution will result in 1024 discrete duty cycles, whereas an 8-bit resolution will result in 256 discrete duty cycles.

The maximum PWM resolution is ten bits when T2PR is 255. The resolution is a function of the T2PR register value as shown by [Equation 23-4](#).

**EQUATION 23-4: PWM RESOLUTION**

$$\text{Resolution} = \frac{\log[4(T2PR + 1)]}{\log(2)} \text{ bits}$$

**Note:** If the pulse-width value is greater than the period, the assigned PWM pin(s) will remain unchanged.

**TABLE 23-2: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS (Fosc = 20 MHz)**

| PWM Frequency             | 1.22 kHz | 4.88 kHz | 19.53 kHz | 78.12 kHz | 156.3 kHz | 208.3 kHz |
|---------------------------|----------|----------|-----------|-----------|-----------|-----------|
| Timer Prescale            | 16       | 4        | 1         | 1         | 1         | 1         |
| T2PR Value                | 0xFF     | 0xFF     | 0xFF      | 0x3F      | 0x1F      | 0x17      |
| Maximum Resolution (bits) | 10       | 10       | 10        | 8         | 7         | 6.6       |

**TABLE 23-3: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS (Fosc = 8 MHz)**

| PWM Frequency             | 1.22 kHz | 4.90 kHz | 19.61 kHz | 76.92 kHz | 153.85 kHz | 200.0 kHz |
|---------------------------|----------|----------|-----------|-----------|------------|-----------|
| Timer Prescale            | 16       | 4        | 1         | 1         | 1          | 1         |
| T2PR Value                | 0x65     | 0x65     | 0x65      | 0x19      | 0x0C       | 0x09      |
| Maximum Resolution (bits) | 8        | 8        | 8         | 6         | 5          | 5         |

## 23.4.7 OPERATION IN SLEEP MODE

In Sleep mode, the T2TMR register will not increment and the state of the module will not change. If the CCPx pin is driving a value, it will continue to drive that value. When the device wakes up, T2TMR will continue from its previous state.

## 23.4.8 CHANGES IN SYSTEM CLOCK FREQUENCY

The PWM frequency is derived from the system clock frequency. Any changes in the system clock frequency will result in changes to the PWM frequency. See [Section 7.0 “Oscillator Module \(with Fail-Safe Clock Monitor\)”](#) for additional details.

## 23.4.9 EFFECTS OF RESET

Any Reset will force all ports to Input mode and the CCP registers to their Reset states.

## 23.5 Register Definitions: CCP Control

Long bit name prefixes for the CCP peripherals are shown below. Refer to [Section 1.3.2.2 “Long Bit Names”](#) for more information.

| Peripheral | Bit Name Prefix |
|------------|-----------------|
| CCP1       | CCP1            |
| CCP2       | CCP2            |
| CCP3       | CCP3            |
| CCP4       | CCP4            |

### REGISTER 23-1: CCPxCON: CCPx CONTROL REGISTER

|         |     |     |         |           |         |         |         |
|---------|-----|-----|---------|-----------|---------|---------|---------|
| R/W-0/0 | U-0 | R-x | R/W-0/0 | R/W-0/0   | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| EN      | —   | OUT | FMT     | MODE<3:0> |         |         |         |
| bit 7   |     |     |         |           |         |         | bit 0   |

#### Legend:

R = Readable bit      W = Writable bit      U = Unimplemented bit, read as '0'  
 -n = Value at POR      '1' = Bit is set      '0' = Bit is cleared      x = Bit is unknown

bit 7      **EN:** CCP Module Enable bit  
           1 = CCP is enabled  
           0 = CCP is disabled

bit 6      **Unimplemented:** Read as '0'

bit 5      **OUT:** CCPx Output Data bit (read-only)

bit 4      **FMT:** CCPW (pulse-width) Alignment bit  
           **MODE = Capture mode:**  
           Unused  
           **MODE = Compare mode:**  
           Unused  
           **MODE = PWM mode:**  
           1 = Left-aligned format  
           0 = Right-aligned format

bit 3-0      **MODE<3:0>:** CCPx Mode Select bits

| MODE | Operating Mode | Operation                                | Set CCPxIF |
|------|----------------|------------------------------------------|------------|
| 11xx | PWM            | PWM operation                            | Yes        |
| 1011 | Compare        | Pulse output; clear TMR1 <sup>(2)</sup>  | Yes        |
| 1010 |                | Pulse output                             | Yes        |
| 1001 |                | Clear output <sup>(1)</sup>              | Yes        |
| 1000 |                | Set output <sup>(1)</sup>                | Yes        |
| 0111 | Capture        | Every 16th rising edge of CCPx input     | Yes        |
| 0110 |                | Every 4th rising edge of CCPx input      | Yes        |
| 0101 |                | Every rising edge of CCPx input          | Yes        |
| 0100 |                | Every falling edge of CCPx input         | Yes        |
| 0011 |                | Every edge of CCPx input                 | Yes        |
| 0010 | Compare        | Toggle output                            | Yes        |
| 0001 |                | Toggle output; clear TMR1 <sup>(2)</sup> | Yes        |
| 0000 | Disabled       |                                          | —          |

**Note 1:** The set and clear operations of the Compare mode are reset by setting MODE = 4'b0000 or EN = 0.  
**Note 2:** When MODE = 0001 or 1011, then the timer associated with the CCP module is cleared. TMR1 is the default selection for the CCP module, so it is used for indication purpose only.



## REGISTER 23-2: CCPTMRS0: CCP TIMERS CONTROL REGISTER 0

| R/W-0/0     | R/W-1/1 | R/W-0/0     | R/W-1/1 | R/W-0/0     | R/W-1/1 | R/W-0/0     | R/W-1/1 |
|-------------|---------|-------------|---------|-------------|---------|-------------|---------|
| C4TSEL<1:0> |         | C3TSEL<1:0> |         | C2TSEL<1:0> |         | C1TSEL<1:0> |         |
| bit 7       |         |             |         |             |         |             | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-6

**C4TSEL<1:0>**: CCP4 Timer Selection bits

- 11 = CCP4 is based off Timer5 in Capture/Compare mode and Timer6 in PWM mode
- 10 = CCP4 is based off Timer3 in Capture/Compare mode and Timer4 in PWM mode
- 01 = CCP4 is based off Timer1 in Capture/Compare mode and Timer2 in PWM mode
- 00 = Reserved

bit 5-4

**C3TSEL<1:0>**: CCP3 Timer Selection bits

- 11 = CCP3 is based off Timer5 in Capture/Compare mode and Timer6 in PWM mode
- 10 = CCP3 is based off Timer3 in Capture/Compare mode and Timer4 in PWM mode
- 01 = CCP3 is based off Timer1 in Capture/Compare mode and Timer2 in PWM mode
- 00 = Reserved

bit 3-2

**C2TSEL<1:0>**: CCP2 Timer Selection bits

- 11 = CCP2 is based off Timer5 in Capture/Compare mode and Timer6 in PWM mode
- 10 = CCP2 is based off Timer3 in Capture/Compare mode and Timer4 in PWM mode
- 01 = CCP2 is based off Timer1 in Capture/Compare mode and Timer2 in PWM mode
- 00 = Reserved

bit 1-0

**C1TSEL<1:0>**: CCP1 Timer Selection bits

- 11 = CCP1 is based off Timer5 in Capture/Compare mode and Timer6 in PWM mode
- 10 = CCP1 is based off Timer3 in Capture/Compare mode and Timer4 in PWM mode
- 01 = CCP1 is based off Timer1 in Capture/Compare mode and Timer2 in PWM mode
- 00 = Reserved

## REGISTER 23-3: CCPTMRS1: CCP TIMERS CONTROL REGISTER 1

| R/W-0/0     | R/W-1/1 | R/W-0/0     | R/W-1/1 | R/W-0/0     | R/W-1/1 | R/W-0/0     | R/W-1/1 |
|-------------|---------|-------------|---------|-------------|---------|-------------|---------|
| P8TSEL<1:0> |         | P7TSEL<1:0> |         | P6TSEL<1:0> |         | P5TSEL<1:0> |         |
| bit 7       |         |             |         |             |         |             | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-6 **P8TSEL<1:0>**: PWM8 Timer Selection bits

11 = PWM8 based on TMR8

10 = PWM8 based on TMR6

01 = PWM8 based on TMR4

00 = PWM8 based on TMR2

bit 5-4 **P7TSEL<1:0>**: PWM7 Timer Selection bits

11 = PWM7 based on TMR8

10 = PWM7 based on TMR6

01 = PWM7 based on TMR4

00 = PWM7 based on TMR2

bit 3-2 **P6TSEL<1:0>**: PWM6 Timer Selection bits

11 = PWM6 based on TMR8

10 = PWM6 based on TMR6

01 = PWM6 based on TMR4

00 = PWM6 based on TMR2

bit 1-0 **P5TSEL<1:0>**: PWM5 Timer Selection bits

11 = PWM5 based on TMR8

10 = PWM5 based on TMR6

01 = PWM5 based on TMR4

00 = PWM5 based on TMR2

## REGISTER 23-4: CCPxCAP: CAPTURE INPUT SELECTION MULTIPLEXER REGISTER

|       |     |     |     |          |         |         |         |
|-------|-----|-----|-----|----------|---------|---------|---------|
| U-0   | U-0 | U-0 | U-0 | R/W-0/x  | R/W-0/x | R/W-0/x | R/W-0/x |
| —     | —   | —   | —   | CTS<3:0> |         |         |         |
| bit 7 |     |     |     | bit 0    |         |         |         |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-3

**Unimplemented:** Read as '0'

bit 2-0

**CTS<2:0>:** Capture Trigger Input Selection bits

| CTS<3:0>  | Connection              |                         |                         |                         |
|-----------|-------------------------|-------------------------|-------------------------|-------------------------|
|           | CCP1                    | CCP2                    | CCP3                    | CCP4                    |
| 1111-1001 | Reserved                |                         |                         |                         |
| 1000      | CAN_rx_timestamp        |                         |                         |                         |
| 0111      | CLC4_out                |                         |                         |                         |
| 0110      | CLC3_out                |                         |                         |                         |
| 0101      | CLC2_out                |                         |                         |                         |
| 0100      | CLC1_out                |                         |                         |                         |
| 0011      | IOC_interrupt           |                         |                         |                         |
| 0010      | CMP2_output             |                         |                         |                         |
| 0001      | CMP1_output             |                         |                         |                         |
| 0000      | Pin selected by CCP1PPS | Pin selected by CCP2PPS | Pin selected by CCP3PPS | Pin selected by CCP4PPS |

## REGISTER 23-5: CCPRxL: CCPx REGISTER LOW BYTE

|         |         |         |         |         |         |         |         |
|---------|---------|---------|---------|---------|---------|---------|---------|
| R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x |
| RL<7:0> |         |         |         |         |         |         |         |
| bit 7   |         |         |         | bit 0   |         |         |         |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-0

MODE = Capture Mode:

**RL<7:0>:** LSB of captured TMR1 value

MODE = Compare Mode:

**RL<7:0>:** LSB compared to TMR1 value

MODE = PWM Mode && FMT = 0:

**RL<7:0>:** CCPW<7:0> – Pulse-Width LS 8 bits

MODE = PWM Mode && FMT = 1:

**RL<7:6>:** CCPW<1:0> – Pulse-Width LS 2 bits

**RL<5:0>:** Not used

**REGISTER 23-6: CCPRxH: CCPx REGISTER HIGH BYTE**

|         |         |         |         |         |         |         |         |
|---------|---------|---------|---------|---------|---------|---------|---------|
| R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x |
| RH<7:0> |         |         |         |         |         |         |         |
| bit 7   |         |         |         | bit 0   |         |         |         |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-0

MODE = Capture Mode:

**RH<7:0>**: MSB of captured TMR1 value

MODE = Compare Mode:

**RH<7:0>**: MSB compared to TMR1 value

MODE = PWM Mode && FMT = 0:

**RH<7:2>**: Not used

**RH<1:0>**: CCPW<9:8> – Pulse-Width MS 2 bits

MODE = PWM Mode && FMT = 1:

**RH<7:0>**: CCPW<9:2> – Pulse-Width MS 8 bits

**TABLE 23-4: SUMMARY OF REGISTERS ASSOCIATED WITH CCPx**

| Name     | Bit 7       | Bit 6 | Bit 5       | Bit 4 | Bit 3       | Bit 2 | Bit 1       | Bit 0 | Register on Page    |
|----------|-------------|-------|-------------|-------|-------------|-------|-------------|-------|---------------------|
| CCPxCON  | EN          | —     | OUT         | FMT   | MODE<3:0>   |       |             |       | <a href="#">336</a> |
| CCPxCAP  | —           | —     | —           | —     | —           | —     | CTS<1:0>    |       | <a href="#">339</a> |
| CCPRxL   | CCPRx<7:0>  |       |             |       |             |       |             |       | <a href="#">339</a> |
| CCPRxH   | CCPRx<15:8> |       |             |       |             |       |             |       | <a href="#">340</a> |
| CCPTMRS0 | C4TSEL<1:0> |       | C3TSEL<1:0> |       | C2TSEL<1:0> |       | C1TSEL<1:0> |       | <a href="#">337</a> |

**Legend:** — = Unimplemented location, read as '0'. Shaded cells are not used by the CCP module.

## 24.0 PULSE-WIDTH MODULATION (PWM)

The PWM module generates a pulse-width modulated signal determined by the duty cycle, period, and resolution that are configured by the following registers:

- TxPR
- TxCON
- PWMxDCH
- PWMxDCL
- PWMxCON

**Note:** The corresponding TRIS bit must be cleared to enable the PWM output on the PWMx pin.

Each PWM module can select the timer source that controls the module. Each module has an independent timer selection which can be accessed using the CCPTMRS1 register ([Register 23-2](#)). Please note that the PWM mode operation is described with respect to T2TMR in the following sections.

[Figure 24-1](#) shows a simplified block diagram of PWM operation.

[Figure 24-2](#) shows a typical waveform of the PWM signal.

**FIGURE 24-1: SIMPLIFIED PWM BLOCK DIAGRAM**

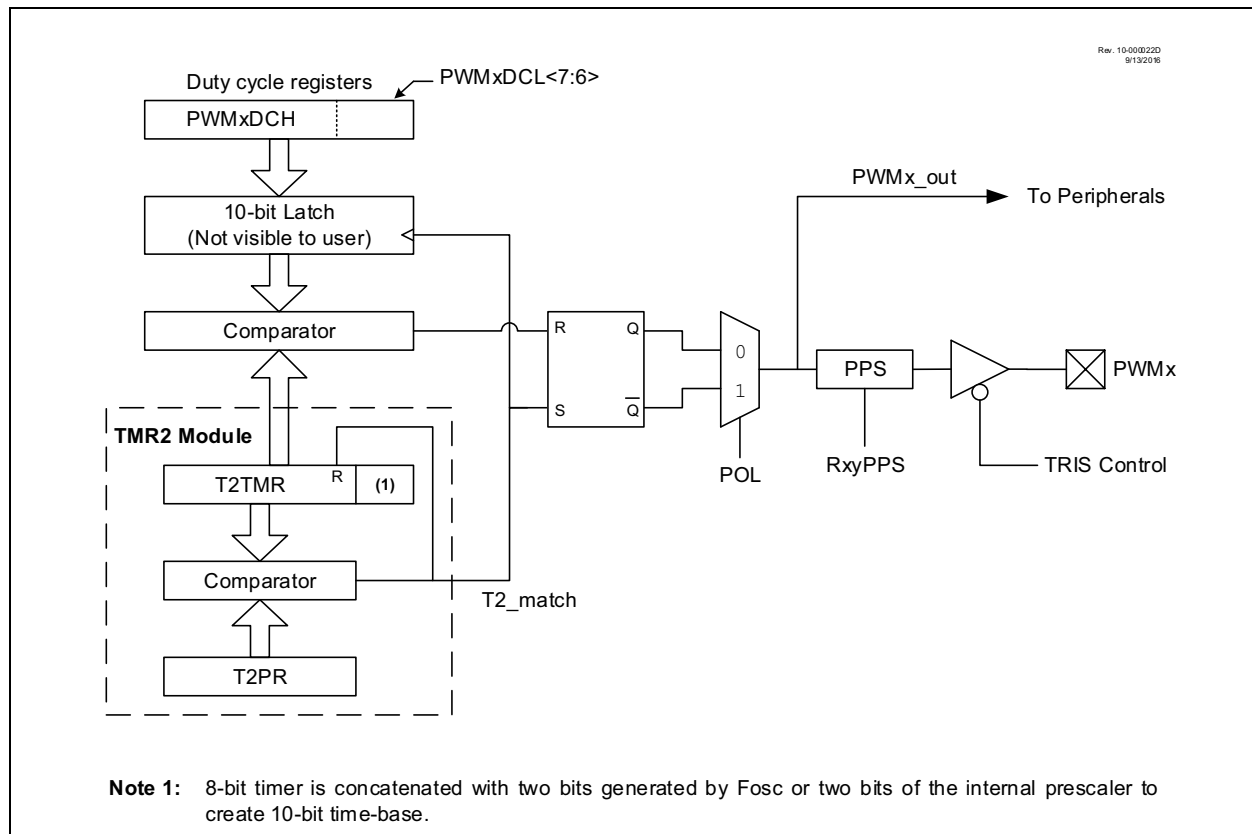
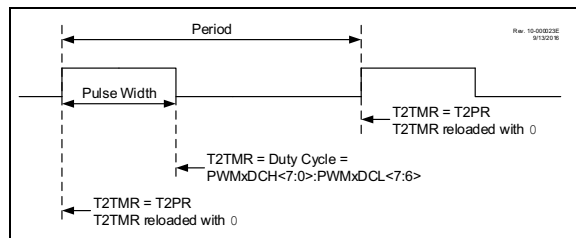


**FIGURE 24-2: PWM OUTPUT**



For a step-by-step procedure on how to set up this module for PWM operation, refer to [Section 24.1.9 "Setup for PWM Operation using PWMx Pins"](#).

## 24.1 PWMx Pin Configuration

All PWM outputs are multiplexed with the PORT data latch. The user must configure the pins as outputs by clearing the associated TRIS bits.

### 24.1.1 FUNDAMENTAL OPERATION

The PWM module produces a 10-bit resolution output. The PWM timer can be selected using the PxTSEL bits in the CCPTMRS1 register. The default selection for PWMx is T2TMR. Please note that the PWM module operation in the following sections is described with respect to T2TMR. Timer2 and T2PR set the period of the PWM. The PWMxDCL and PWMxDCH registers configure the duty cycle. The period is common to all PWM modules, whereas the duty cycle is independently controlled.

**Note:** The Timer2 postscaler is not used in the determination of the PWM frequency. The postscaler could be used to have a servo update rate at a different frequency than the PWM output.

All PWM outputs associated with Timer2 are set when T2TMR is cleared. Each PWMx is cleared when T2TMR is equal to the value specified in the corresponding PWMxDCH (8 MSb) and PWMxDCL<7:6> (2 LSb) registers. When the value is greater than or equal to T2PR, the PWM output is never cleared (100% duty cycle).

**Note:** The PWMxDCH and PWMxDCL registers are double buffered. The buffers are updated when Timer2 matches T2PR. Care should be taken to update both registers before the timer match occurs.

### 24.1.2 PWM OUTPUT POLARITY

The output polarity is inverted by setting the PWMxPOL bit of the PWMxCON register.

### 24.1.3 PWM PERIOD

The PWM period is specified by the T2PR register of Timer2. The PWM period can be calculated using the formula of Equation 24-1. It is required to have FOSC/4 as clock input to Timer2/4/6 for correct PWM operation.

#### EQUATION 24-1: PWM PERIOD

$$\text{PWM Period} = [(T2PR + 1) \cdot 4 \cdot T_{osc}] \cdot (\text{TM R2 Prescale Value})$$

**Note:**  $T_{osc} = 1/F_{osc}$

When T2TMR is equal to T2PR, the following three events occur on the next increment cycle:

- T2TMR is cleared
- The PWM output is active. (Exception: When the PWM duty cycle = 0%, the PWM output will remain inactive.)
- The PWMxDCH and PWMxDCL register values are latched into the buffers.

**Note:** The Timer2 postscaler has no effect on the PWM operation.

### 24.1.4 PWM DUTY CYCLE

The PWM duty cycle is specified by writing a 10-bit value to the PWMxDCH and PWMxDCL register pair. The PWMxDCH register contains the eight MSbs and the PWMxDCL<7:6>, the two LSbs. The PWMxDCH and PWMxDCL registers can be written to at any time.

Equation 24-2 is used to calculate the PWM pulse width.

Equation 24-3 is used to calculate the PWM duty cycle ratio.

#### EQUATION 24-2: PULSE WIDTH

$$\text{Pulse Width} = (\text{PWMxDCH} : \text{PWMxDCL} < 7:6 >) \cdot T_{osc} \cdot (\text{TM R2 Prescale Value})$$

**Note:**  $T_{osc} = 1/F_{osc}$

#### EQUATION 24-3: DUTY CYCLE RATIO

$$\text{Duty Cycle Ratio} = \frac{(\text{PWMxDCH} : \text{PWMxDCL} < 7:6 >)}{4(T2PR + 1)}$$

The 8-bit timer T2TMR register is concatenated with the two Least Significant bits of 1/FOSC, adjusted by the Timer2 prescaler to create the 10-bit time base. The system clock is used if the Timer2 prescaler is set to 1:1.

## 24.1.5 PWM RESOLUTION

The resolution determines the number of available duty cycles for a given period. For example, a 10-bit resolution will result in 1024 discrete duty cycles, whereas an 8-bit resolution will result in 256 discrete duty cycles.

The maximum PWM resolution is ten bits when T2PR is 255. The resolution is a function of the T2PR register value as shown by [Equation 24-4](#).

### EQUATION 24-4: PWM RESOLUTION

$$\text{Resolution} = \frac{\log[4(\text{T2PR} + 1)]}{\log(2)} \text{ bits}$$

**Note:** If the pulse-width value is greater than the period, the assigned PWM pin(s) will remain unchanged.

**TABLE 24-1: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS (Fosc = 20 MHz)**

| PWM Frequency             | 0.31 kHz | 4.88 kHz | 19.53 kHz | 78.12 kHz | 156.3 kHz | 208.3 kHz |
|---------------------------|----------|----------|-----------|-----------|-----------|-----------|
| Timer Prescale            | 64       | 4        | 1         | 1         | 1         | 1         |
| T2PR Value                | 0xFF     | 0xFF     | 0xFF      | 0x3F      | 0x1F      | 0x17      |
| Maximum Resolution (bits) | 10       | 10       | 10        | 8         | 7         | 6.6       |

**TABLE 24-2: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS (Fosc = 8 MHz)**

| PWM Frequency             | 0.31 kHz | 4.90 kHz | 19.61 kHz | 76.92 kHz | 153.85 kHz | 200.0 kHz |
|---------------------------|----------|----------|-----------|-----------|------------|-----------|
| Timer Prescale            | 64       | 4        | 1         | 1         | 1          | 1         |
| T2PR Value                | 0x65     | 0x65     | 0x65      | 0x19      | 0x0C       | 0x09      |
| Maximum Resolution (bits) | 8        | 8        | 8         | 6         | 5          | 5         |

## 24.1.6 OPERATION IN SLEEP MODE

In Sleep mode, the T2TMR register will not increment and the state of the module will not change. If the PWMx pin is driving a value, it will continue to drive that value. When the device wakes up, T2TMR will continue from its previous state.

## 24.1.7 CHANGES IN SYSTEM CLOCK FREQUENCY

The PWM frequency is derived from the system clock frequency (Fosc). Any changes in the system clock frequency will result in changes to the PWM frequency. Refer to [Section 7.0 “Oscillator Module \(with Fail-Safe Clock Monitor\)”](#) for additional details.

## 24.1.8 EFFECTS OF RESET

Any Reset will force all ports to Input mode and the PWM registers to their Reset states.

## 24.1.9 SETUP FOR PWM OPERATION USING PWMx PINS

The following steps should be taken when configuring the module for PWM operation using the PWMx pins:

1. Disable the PWMx pin output driver(s) by setting the associated TRIS bit(s).
2. Clear the PWMxCON register.
3. Load the T2PR register with the PWM period value.
4. Load the PWMxDCH register and bits <7:6> of the PWMxDCL register with the PWM duty cycle value.
5. Configure and start Timer2:
  - Clear the TMR2IF interrupt flag bit of the respective PIR register. See Note 1 below.
  - Select the timer clock source to be as  $F_{osc}/4$  using the TxCLK register. This is required for correct operation of the PWM module.
  - Configure the CKPS bits of the T2CON register with the Timer2 prescale value.
  - Enable Timer2 by setting the ON bit of the T2CON register.
6. Enable PWM output pin and wait until Timer2 overflows, TMR2IF bit of the respective PIR register is set. See note below.
7. Enable the PWMx pin output driver(s) by clearing the associated TRIS bit(s) and setting the desired pin PPS control bits.
8. Configure the PWM module by loading the PWMxCON register with the appropriate values.

**Note 1:** In order to send a complete duty cycle and period on the first PWM output, the above steps must be followed in the order given. If it is not critical to start with a complete PWM signal, then move Step 8 to replace Step 4.

**2:** For operation with other peripherals only, disable PWMx pin outputs.

## 24.1.10 SETUP FOR PWM OPERATION TO OTHER DEVICE PERIPHERALS

The following steps should be taken when configuring the module for PWM operation to be used by other device peripherals:

1. Disable the PWMx pin output driver(s) by setting the associated TRIS bit(s).
2. Clear the PWMxCON register.
3. Load the T2PR register with the PWM period value.
4. Load the PWMxDCH register and bits <7:6> of the PWMxDCL register with the PWM duty cycle value.
5. Configure and start Timer2:
  - Clear the TMR2IF interrupt flag bit of the respective PIR register. See Note 1 below.
  - Select the timer clock source to be as  $F_{osc}/4$  using the TxCLK register. This is required for correct operation of the PWM module.
  - Configure the CKPS bits of the T2CON register with the Timer2 prescale value.
  - Enable Timer2 by setting the ON bit of the T2CON register.
6. Enable PWM output pin:
  - Wait until Timer2 overflows, TMR2IF bit of the respective PIR register is set. See Note 1 below.
7. Configure the PWM module by loading the PWMxCON register with the appropriate values.

**Note 1:** In order to send a complete duty cycle and period on the first PWM output, the above steps must be included in the setup sequence. If it is not critical to start with a complete PWM signal on the first output, then step 6 may be ignored.



## 24.2 Register Definitions: PWM Control

Long bit name prefixes for the PWM peripherals are shown below. Refer to [Section 1.3.2.2 “Long Bit Names”](#) for more information.

| Peripheral | Bit Name Prefix |
|------------|-----------------|
| PWM3       | PWM3            |
| PWM4       | PWM4            |

### REGISTER 24-1: PWMxCON: PWM CONTROL REGISTER

|         |     |       |         |     |     |     |       |
|---------|-----|-------|---------|-----|-----|-----|-------|
| R/W-0/0 | U-0 | R-0/0 | R/W-0/0 | U-0 | U-0 | U-0 | U-0   |
| EN      | —   | OUT   | POL     | —   | —   | —   | —     |
| bit 7   |     |       |         |     |     |     | bit 0 |

#### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as ‘0’

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

‘1’ = Bit is set

‘0’ = Bit is cleared

- bit 7 **EN:** PWM Module Enable bit  
1 = PWM module is enabled  
0 = PWM module is disabled
- bit 6 **Unimplemented:** Read as ‘0’
- bit 5 **OUT:** PWM Module Output Level When Bit is Read
- bit 4 **POL:** PWM Output Polarity Select bit  
1 = PWM output is inverted  
0 = PWM output is normal
- bit 3-0 **Unimplemented:** Read as ‘0’

## REGISTER 24-2: CCPTMRS1: CCP TIMERS CONTROL REGISTER 1

|             |         |             |         |             |         |             |         |
|-------------|---------|-------------|---------|-------------|---------|-------------|---------|
| R/W-0/0     | R/W-1/1 | R/W-0/0     | R/W-1/1 | R/W-0/0     | R/W-1/1 | R/W-0/0     | R/W-1/1 |
| P8TSEL<1:0> |         | P7TSEL<1:0> |         | P6TSEL<1:0> |         | P5TSEL<1:0> |         |
| bit 7       |         |             |         |             |         |             | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-6 **P8TSEL<1:0>**: PWM8 Timer Selection bits

11 = PWM8 based on TMR6

10 = PWM8 based on TMR4

01 = PWM8 based on TMR2

00 = Reserved

bit 5-4 **P7TSEL<1:0>**: PWM7 Timer Selection bits

11 = PWM7 based on TMR6

10 = PWM7 based on TMR4

01 = PWM7 based on TMR2

00 = Reserved

bit 3-2 **P6TSEL<1:0>**: PWM6 Timer Selection bits

11 = PWM6 based on TMR6

10 = PWM6 based on TMR4

01 = PWM6 based on TMR2

00 = Reserved

bit 1-0 **P5TSEL<1:0>**: PWM5 Timer Selection bits

11 = PWM5 based on TMR6

10 = PWM5 based on TMR4

01 = PWM5 based on TMR2

00 = Reserved

**REGISTER 24-3: PWMxDCH: PWM DUTY CYCLE HIGH BITS**

|         |         |         |         |         |         |         |         |
|---------|---------|---------|---------|---------|---------|---------|---------|
| R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
| DC<9:2> |         |         |         |         |         |         |         |
| bit 7   |         |         |         |         |         |         |         |
| bit 0   |         |         |         |         |         |         |         |

**Legend:**

R = Readable bit      W = Writable bit      U = Unimplemented bit, read as '0'  
u = Bit is unchanged      x = Bit is unknown      -n/n = Value at POR and BOR/Value at all other Resets  
'1' = Bit is set      '0' = Bit is cleared

bit 7-0      **DC<9:2>**: PWM Duty Cycle Most Significant bits  
These bits are the MSBs of the PWM duty cycle. The two LSbs are found in PWMxDCL Register.

**REGISTER 24-4: PWMxDCL: PWM DUTY CYCLE LOW BITS**

|         |         |     |     |     |     |     |     |
|---------|---------|-----|-----|-----|-----|-----|-----|
| R/W-x/u | R/W-x/u | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
| DC<1:0> |         | —   | —   | —   | —   | —   | —   |
| bit 7   |         |     |     |     |     |     |     |
| bit 0   |         |     |     |     |     |     |     |

**Legend:**

R = Readable bit      W = Writable bit      U = Unimplemented bit, read as '0'  
u = Bit is unchanged      x = Bit is unknown      -n/n = Value at POR and BOR/Value at all other Resets  
'1' = Bit is set      '0' = Bit is cleared

bit 7-6      **DC<1:0>**: PWM Duty Cycle Least Significant bits  
These bits are the LSbs of the PWM duty cycle. The MSBs are found in PWMxDCH Register.

bit 5-0      **Unimplemented**: Read as '0'

**TABLE 24-3: SUMMARY OF REGISTERS ASSOCIATED WITH PWM**

| Name     | Bit 7       | Bit 6 | Bit 5       | Bit 4 | Bit 3       | Bit 2 | Bit 1       | Bit 0 | Register on Page |
|----------|-------------|-------|-------------|-------|-------------|-------|-------------|-------|------------------|
| PWMxCON  | EN          | —     | OUT         | POL   | —           | —     | —           | —     | 345              |
| PWMxDCH  | DC<9:2>     |       |             |       |             |       |             |       | 347              |
| PWMxDCL  | DC<1:0>     |       | —           | —     | —           | —     | —           | —     | 347              |
| CCPTMRS1 | P8TSEL<1:0> |       | P7TSEL<1:0> |       | P6TSEL<1:0> |       | P5TSEL<1:0> |       | 346              |

**Legend:** — = Unimplemented locations, read as '0', u = unchanged, x = unknown. Shaded cells are not used by the PWM.

## 25.0 SIGNAL MEASUREMENT TIMER (SMTx)

The SMT is a 24-bit counter with advanced clock and gating logic, which can be configured for measuring a variety of digital signal parameters such as pulse width, frequency and duty cycle, and the time difference between edges on two signals. The device has only one SMT module implemented.

Features of the SMT include:

- 24-bit timer/counter
  - Three 8-bit registers (SMTxL/H/U)
  - Readable and writable
  - Optional 16-bit operating mode
- Two 24-bit measurement capture registers
- One 24-bit period match register
- Multi-mode operation, including relative timing measurement
- Interrupt on period match
- Multiple clock, gate and signal sources
- Interrupt on acquisition complete
- Ability to read current input values

**FIGURE 25-1: SMT BLOCK DIAGRAM**

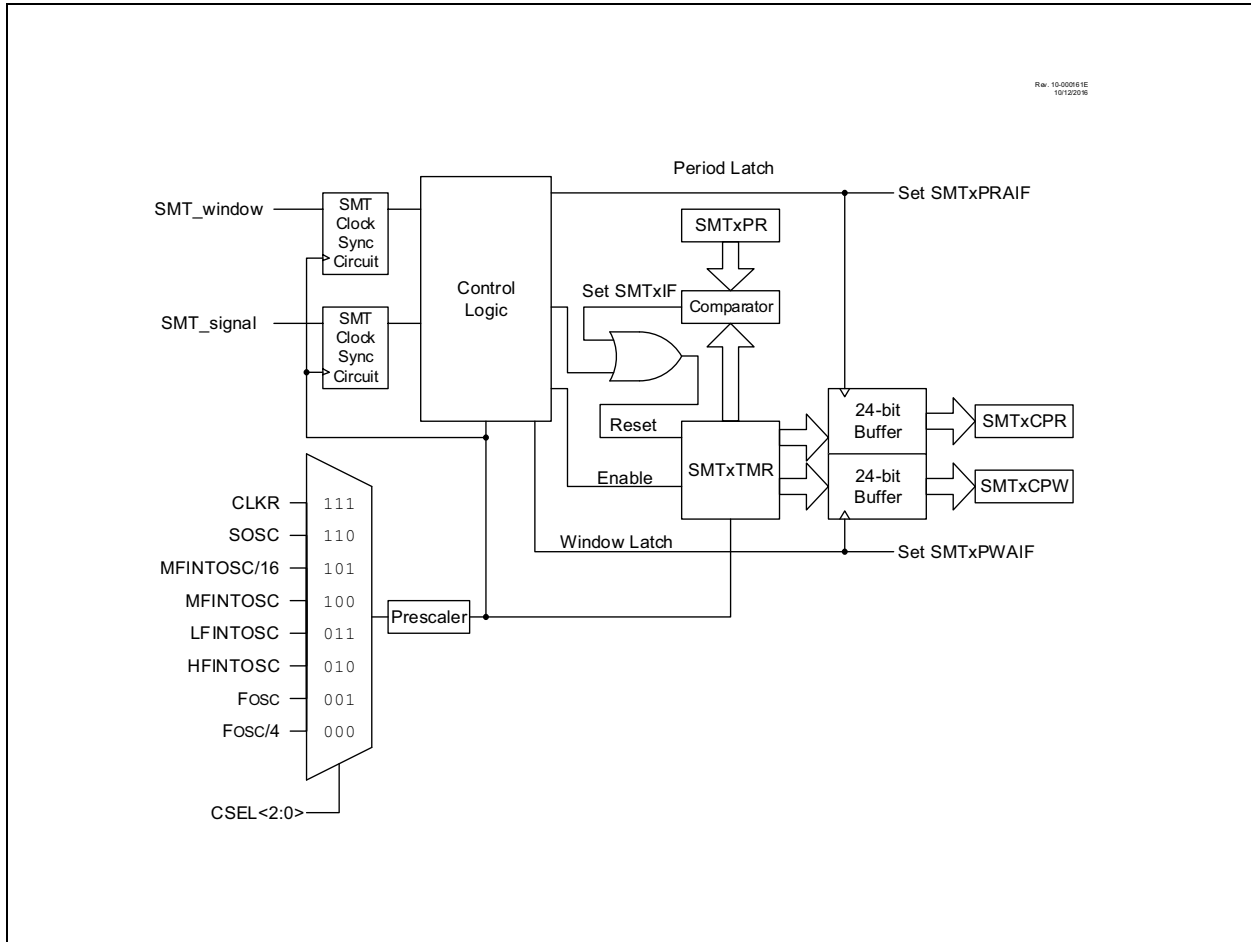
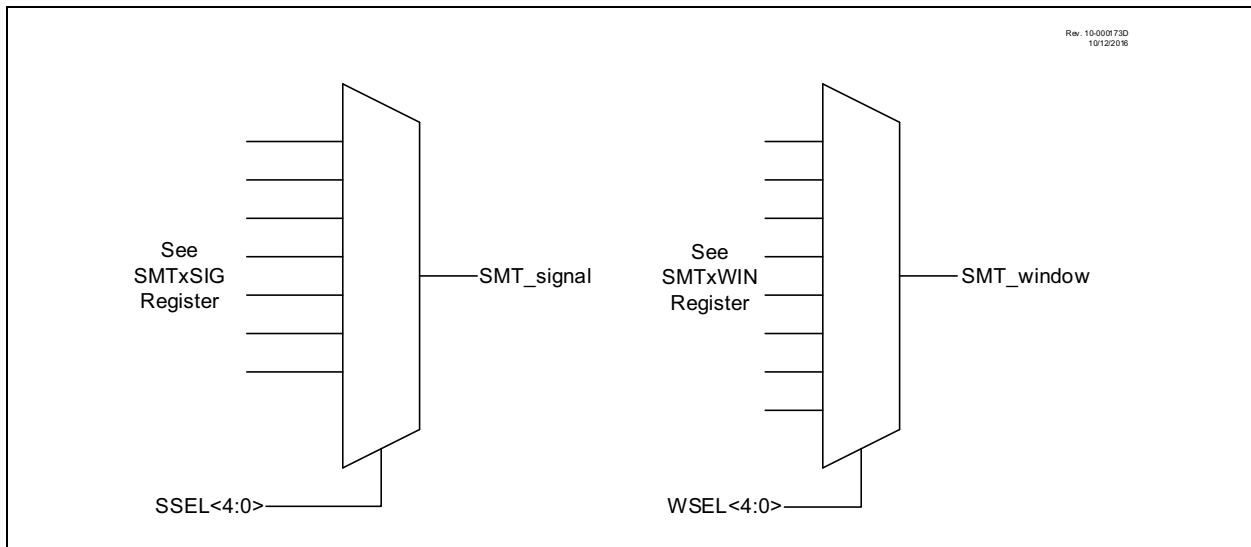


**FIGURE 25-2: SMT SIGNAL AND WINDOW BLOCK DIAGRAM**



## 25.1 SMT Operation

The core of the module is the 24-bit counter, SMTxTMR combined with a complex data acquisition front-end. Depending on the mode of operation selected, the SMT can perform a variety of measurements summarized in [Table 25-1](#).

### 25.1.1 CLOCK SOURCES

Clock sources available to the SMT include:

- Fosc
- Fosc/4
- HFINTOSC 16 MHz
- LFINTOSC
- MFINTOSC 31.25 kHz

The SMT clock source is selected by configuring the CSEL<2:0> bits in the SMTxCLK register. The clock source can also be prescaled using the PS<1:0> bits of the SMTxCON0 register. The prescaled clock source is used to clock both the counter and any synchronization logic used by the module.

### 25.1.2 PERIOD MATCH INTERRUPT

Similar to other timers, the SMT triggers an interrupt when SMTxTMR rolls over to '0'. This happens when SMTxTMR = SMTxPR, regardless of mode. Hence, in any mode that relies on an external signal or a window to reset the timer, proper operation requires that SMTxPR be set to a period larger than that of the expected signal or window.

## 25.2 Basic Timer Function Registers

The SMTxTMR time base and the SMTxCPW/ SMTxPR/SMTxCPR buffer registers serve several functions and can be manually updated using software.

### 25.2.1 TIME BASE

The SMTxTMR is the 24-bit counter that is the center of the SMT. It is used as the basic counter/timer for measurement in each of the modes of the SMT. It can be reset to a value of 24'h00\_0000 by setting the RST bit of the SMTxSTAT register. It can be written to and read from software, but it is not guarded for atomic access, therefore reads and writes to the SMTxTMR should only be made when the GO = 0, or the software should have other measures to ensure integrity of SMTxTMR reads/writes.

### 25.2.2 PULSE-WIDTH LATCH REGISTERS

The SMTxCPW registers are the 24-bit SMT pulse-width latch. They are used to latch in the value of the SMTxTMR when triggered by various signals, which are determined by the mode the SMT is currently in. The SMTxCPW registers can also be updated with the current value of the SMTxTMR value by setting the CPWUP bit of the SMTxSTAT register.

### 25.2.3 PERIOD LATCH REGISTERS

The SMTxCPR registers are the 24-bit SMT period latch. They are used to latch in other values of the SMTxTMR when triggered by various other signals, which are determined by the mode the SMT is currently in.

The SMTxCPR registers can also be updated with the current value of the SMTxTMR value by setting the CPRUP bit in the SMTxSTAT register.

## 25.3 Halt Operation

The counter can be prevented from rolling-over using the STP bit in the SMTxCON0 register. When halting is enabled, the period match interrupt persists until the SMTxTMR is reset (either by a manual Reset, [Section 25.2.1 "Time Base"](#)) or by clearing the GO bit of the SMTxCON1 register and writing the SMTxTMR values in software.

## 25.4 Polarity Control

The three input signals for the SMT have polarity control to determine whether or not they are active-high/positive edge or active-low/negative edge signals.

The following bits apply to Polarity Control:

- WSEL bit (Window Polarity)
- SSEL bit (Signal Polarity)
- CSEL bit (Clock Polarity)

These bits are located in the SMTxCON0 register.

## 25.5 Status Information

The SMT provides input status information for the user without requiring the need to deal with the polarity of the incoming signals.

### 25.5.1 WINDOW STATUS

Window status is determined by the WS bit of the SMTxSTAT register. This bit is only used in Windowed Measure, Gated Counter and Gated Window Measure modes, and is only valid when TS = 1, and will be delayed in time by synchronizer delays in non-Counter modes.

### 25.5.2 SIGNAL STATUS

Signal status is determined by the AS bit of the SMTxSTAT register. This bit is used in all modes except Window Measure, Time of Flight and Capture modes, and is only valid when TS = 1, and will be delayed in time by synchronizer delays in non-Counter modes.

### 25.5.3 GO STATUS

Timer run status is determined by the TS bit of the SMTxSTAT register, and will be delayed in time by synchronizer delays in non-Counter modes.

## 25.6 Modes of Operation

The modes of operation are summarized in [Table 25-1](#). The following sections provide detailed descriptions, examples of how the modes can be used. Note that all waveforms assume WPOL/SPOL/CPOL = 0. When WPOL/SPOL/CPOL = 1, all SMTSIGx, SMTWINx and SMT clock signals will have a polarity opposite to that indicated. For all modes, the REPEAT bit controls whether the acquisition is repeated or single. When REPEAT = 0 (Single Acquisition mode), the timer will stop incrementing and the GO bit will be reset upon the completion of an acquisition. Otherwise, the timer will continue and allow for continued acquisitions to overwrite the previous ones until the timer is stopped in software.

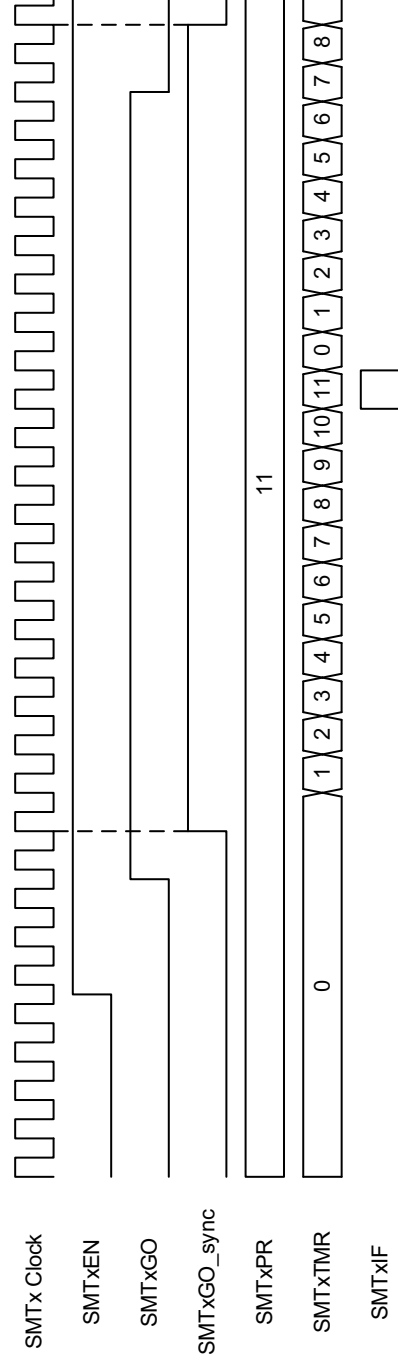
### 25.6.1 TIMER MODE

Timer mode is the simplest mode of operation where the SMTxTMR is used as a 16/24-bit timer. No data acquisition takes place in this mode. The timer increments as long as the GO bit has been set by software. No SMT window or SMT signal events affect the GO bit. Everything is synchronized to the SMT clock source. When the timer experiences a period match (SMTxTMR = SMTxPR), SMTxTMR is reset and the period match interrupt trips. See [Figure 25-3](#).

**TABLE 25-1: MODES OF OPERATION**

| MODE      | Mode of Operation                 | Synchronous Operation | Reference                                                    |
|-----------|-----------------------------------|-----------------------|--------------------------------------------------------------|
| 0000      | Timer                             | Yes                   | <a href="#">Section 25.6.1 “Timer Mode”</a>                  |
| 0001      | Gated Timer                       | Yes                   | <a href="#">Section 25.6.2 “Gated Timer Mode”</a>            |
| 0010      | Period and Duty Cycle Acquisition | Yes                   | <a href="#">Section 25.6.3 “Period and Duty Cycle Mode”</a>  |
| 0011      | High and Low Time Measurement     | Yes                   | <a href="#">Section 25.6.4 “High and Low Measure Mode”</a>   |
| 0100      | Windowed Measurement              | Yes                   | <a href="#">Section 25.6.5 “Windowed Measure Mode”</a>       |
| 0101      | Gated Windowed Measurement        | Yes                   | <a href="#">Section 25.6.6 “Gated Windowed Measure Mode”</a> |
| 0110      | Time of Flight                    | Yes                   | <a href="#">Section 25.6.7 “Time of Flight Measure Mode”</a> |
| 0111      | Capture                           | Yes                   | <a href="#">Section 25.6.8 “Capture Mode”</a>                |
| 1000      | Counter                           | No                    | <a href="#">Section 25.6.9 “Counter Mode”</a>                |
| 1001      | Gated Counter                     | No                    | <a href="#">Section 25.6.10 “Gated Counter Mode”</a>         |
| 1010      | Windowed Counter                  | No                    | <a href="#">Section 25.6.11 “Windowed Counter Mode”</a>      |
| 1011–1111 | Reserved                          | —                     | —                                                            |

**FIGURE 25-3: TIMER MODE TIMING DIAGRAM**





## 25.6.2 GATED TIMER MODE

Gated Timer mode uses the SMTSIGx input to control whether or not the SMTxTMR will increment. Upon a falling edge of the external signal, the SMTxCPW register will update to the current value of the SMTxTMR. Example waveforms for both repeated and single acquisitions are provided in [Figure 25-4](#) and [Figure 25-5](#).

**FIGURE 25-4: GATED TIMER MODE REPEAT ACQUISITION TIMING DIAGRAM**

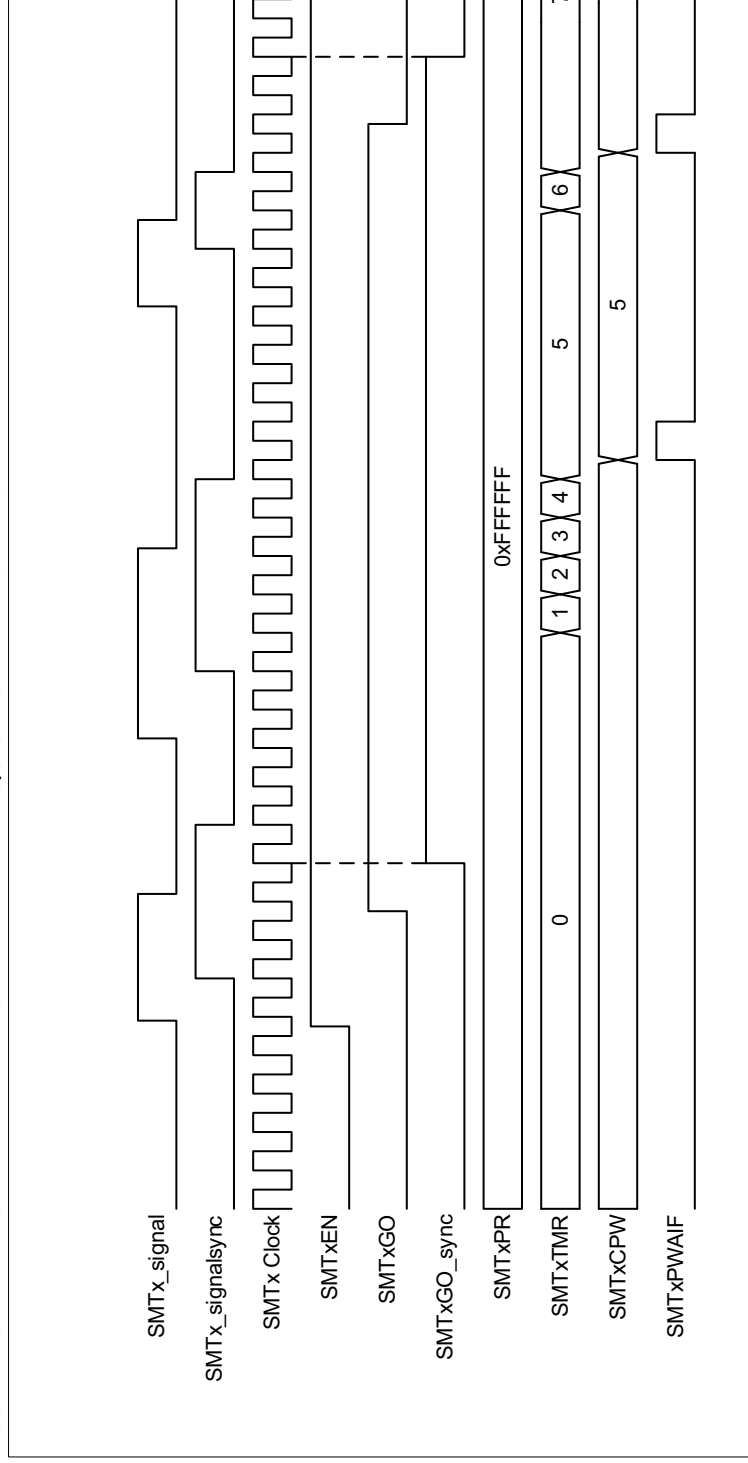
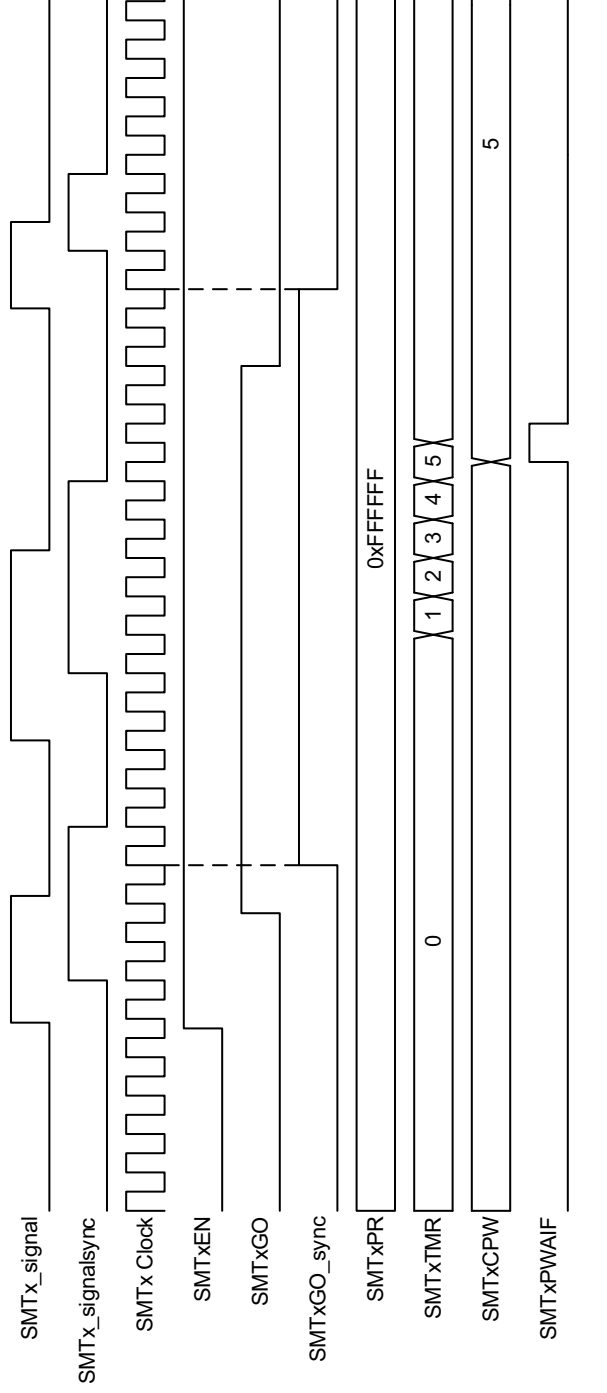


**FIGURE 25-5: GATED TIMER MODE SINGLE ACQUISITION TIMING DIAGRAM**



## 25.6.3 PERIOD AND DUTY CYCLE MODE

In Duty Cycle mode, either the duty cycle or period (depending on polarity) of the SMTx\_signal can be acquired relative to the SMT clock. The CPW register is updated on a falling edge of the signal, and the CPR register is updated on a rising edge of the signal, along with the SMTxTMR resetting to 0x0001. In addition, the GO bit is reset on a rising edge when the SMT is in Single Acquisition mode. See [Figure 25-6](#) and [Figure 25-7](#).

**FIGURE 25-6: PERIOD AND DUTY-CYCLE REPEAT ACQUISITION MODE TIMING DIAGRAM**

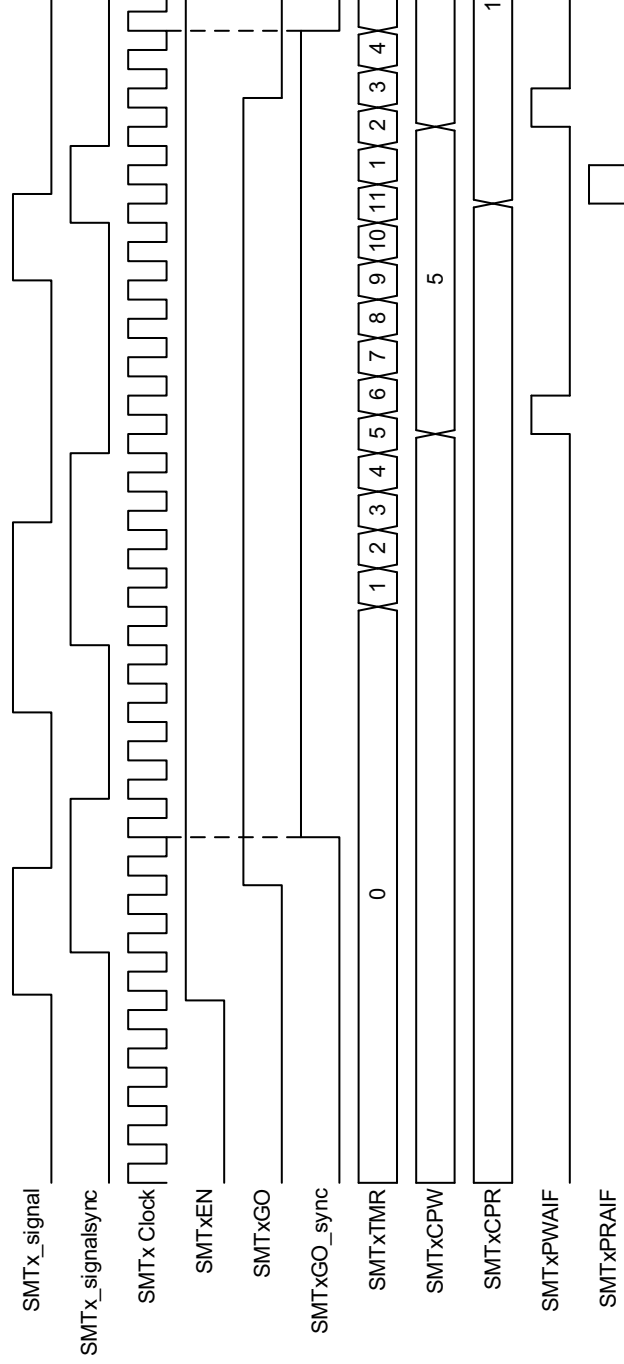
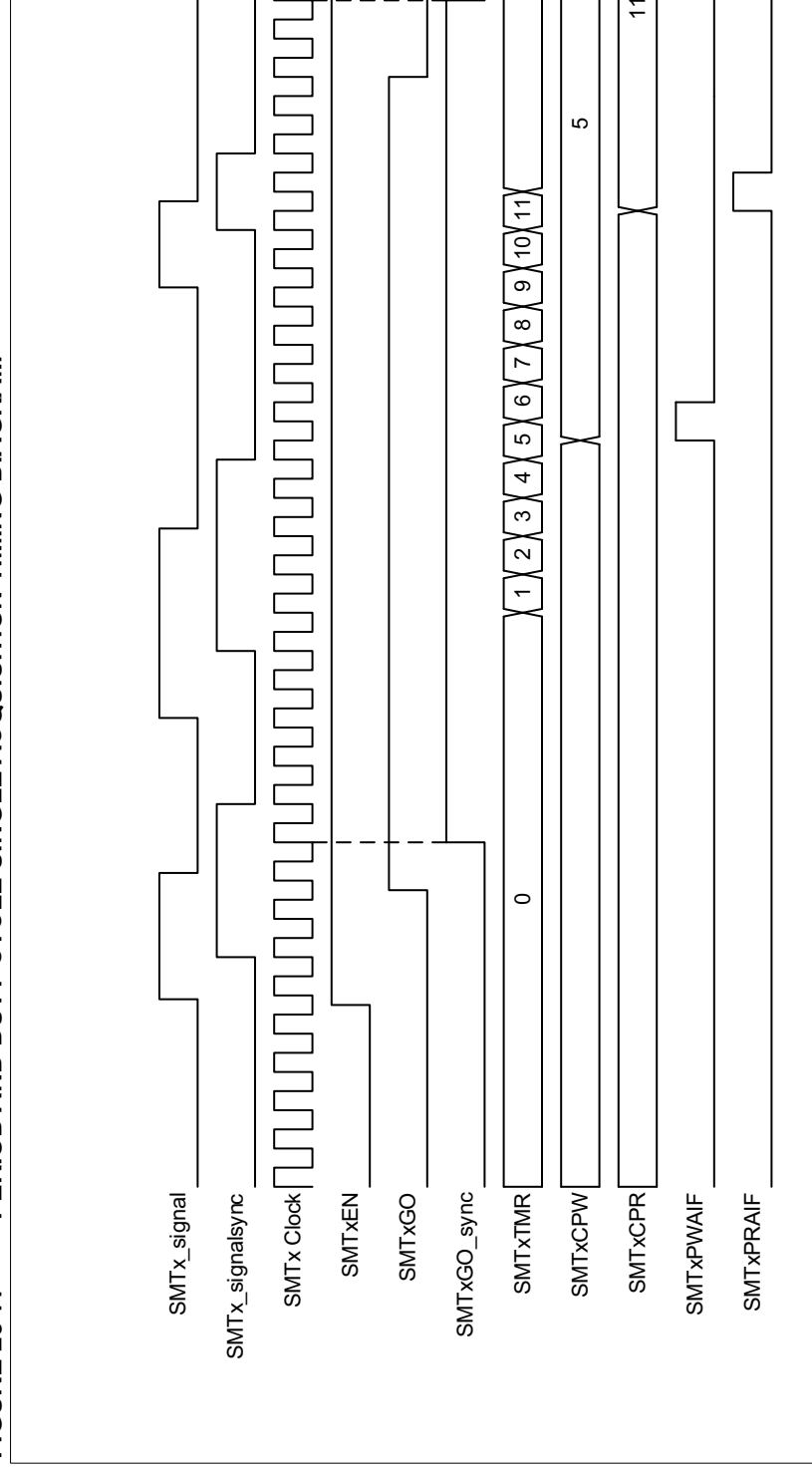


**FIGURE 25-7: PERIOD AND DUTY-CYCLE SINGLE ACQUISITION TIMING DIAGRAM**



## 25.6.4 HIGH AND LOW MEASURE MODE

This mode measures the high and low pulse time of the SMTSIGx relative to the SMT clock. It begins incrementing the SMTxTMR on a rising edge on the SMTSIGx input, then updates the SMTxCPW register with the value and resets the SMTxTMR on a falling edge, starting to increment again. Upon observing another rising edge, it updates the SMTxCPR register with its current value and once again resets the SMTxTMR value and begins incrementing again. See [Figure 25-8](#) and [Figure 25-9](#).

**FIGURE 25-8: HIGH AND LOW MEASURE MODE REPEAT ACQUISITION TIMING DIAGRAM**

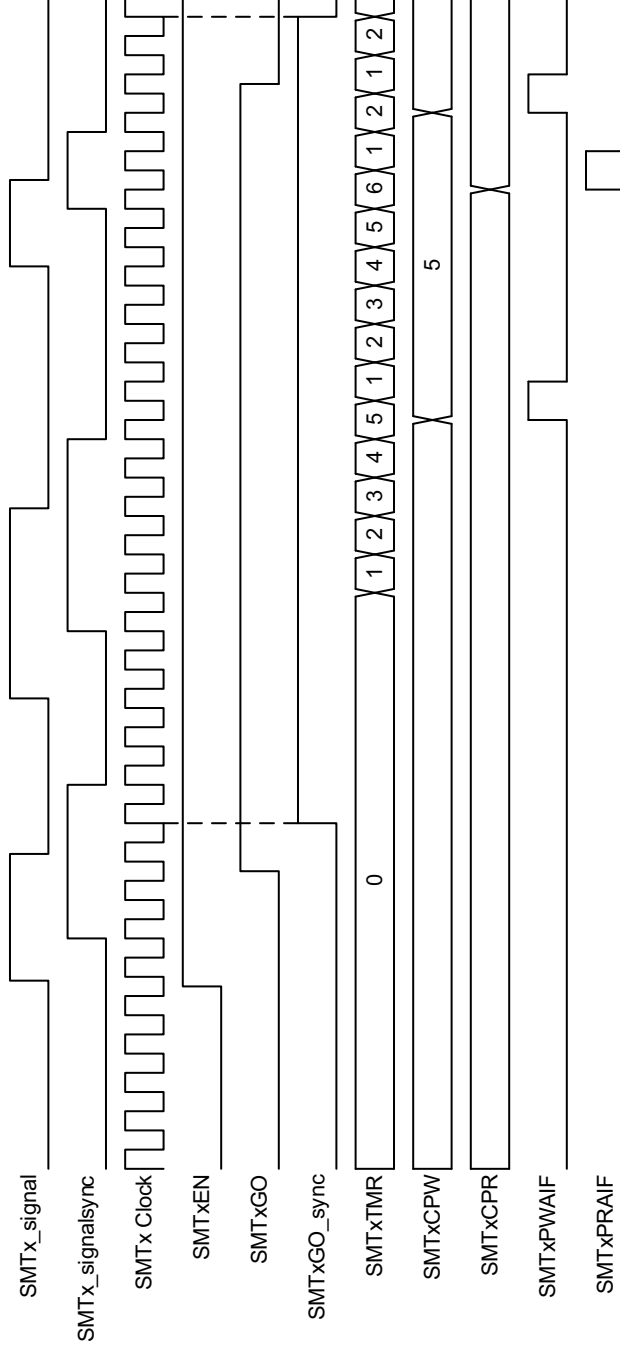
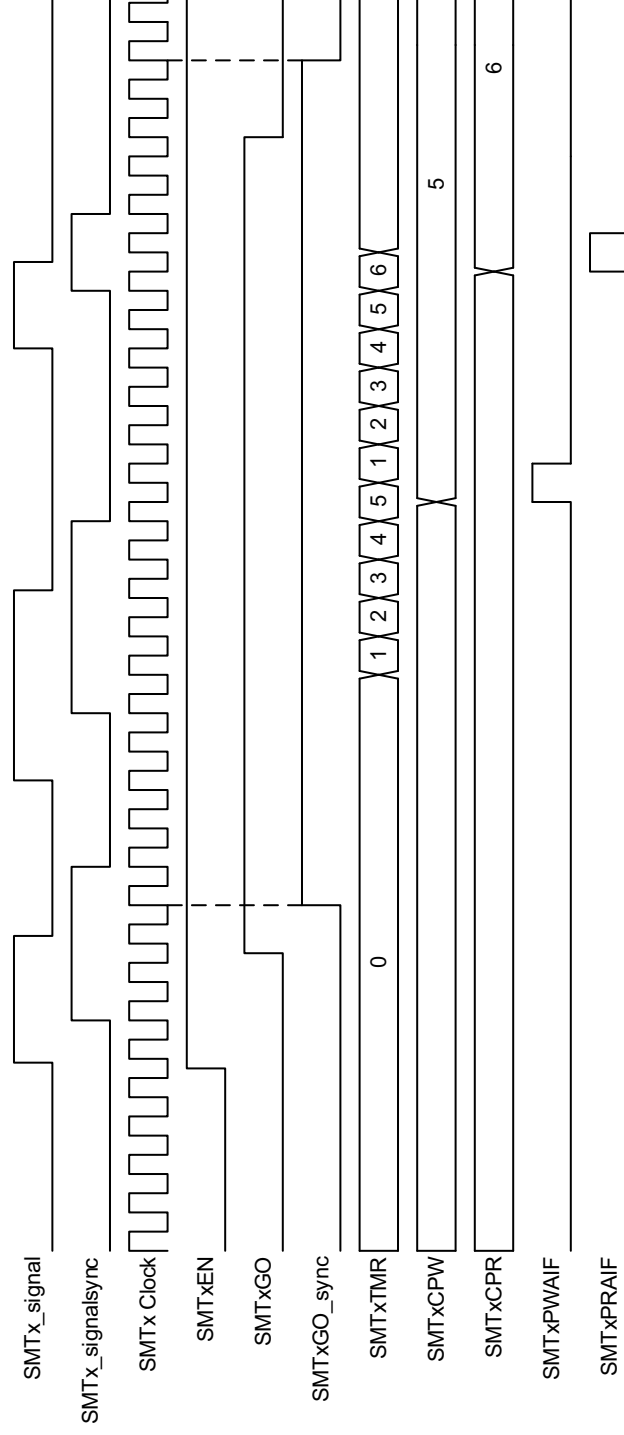




**FIGURE 25-9: HIGH AND LOW MEASURE MODE SINGLE ACQUISITION TIMING DIAGRAM**



## 25.6.5 WINDOWED MEASURE MODE

This mode measures the window duration of the SMTWINx input of the SMT. It begins incrementing the timer on a rising edge of the SMTWINx input and updates the SMTxCPR register with the value of the timer and resets the timer on a second rising edge. See [Figure 25-10](#) and [Figure 25-11](#).

**FIGURE 25-10: WINDOWED MEASURE MODE REPEAT ACQUISITION TIMING DIAGRAM**

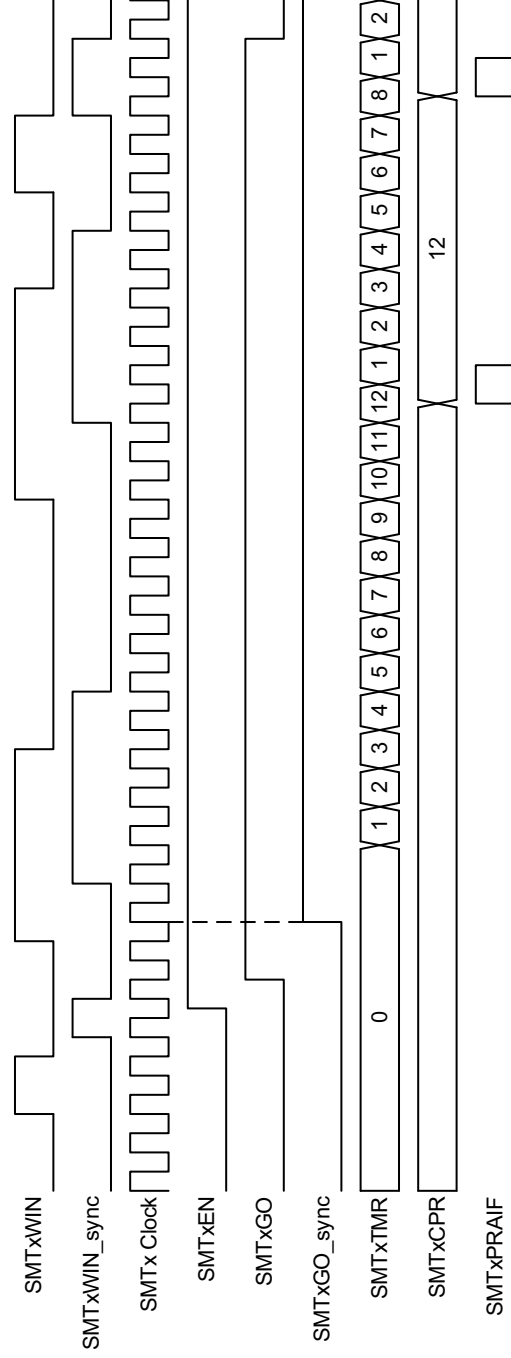
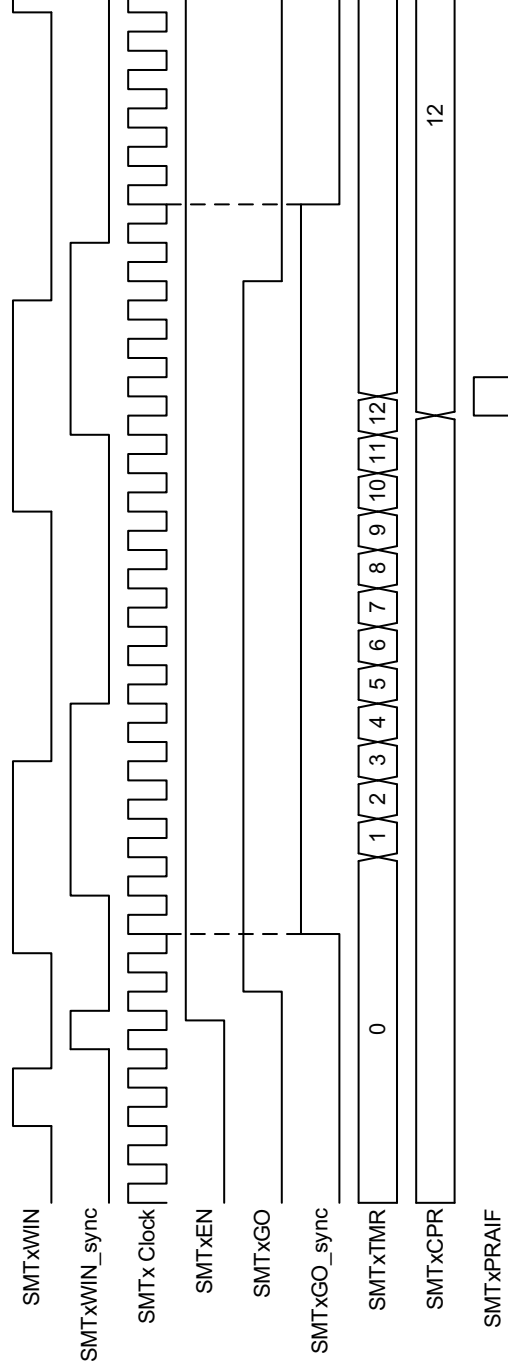


**FIGURE 25-11: WINDOWED MEASURE MODE SINGLE ACQUISITION TIMING DIAGRAM**



## 25.6.6 GATED WINDOWED MEASURE MODE

This mode measures the duty cycle of the SMTx\_signal input over a known input window. It does so by incrementing the timer on each pulse of the clock signal while the SMTx\_signal input is high, updating the SMTxCPR register and resetting the timer on every rising edge of the SMTWINx input after the first. See [Figure 25-12](#) and [Figure 25-13](#).

**FIGURE 25-12: GATED WINDOWED MEASURE MODE REPEAT ACQUISITION TIMING DIAGRAM**

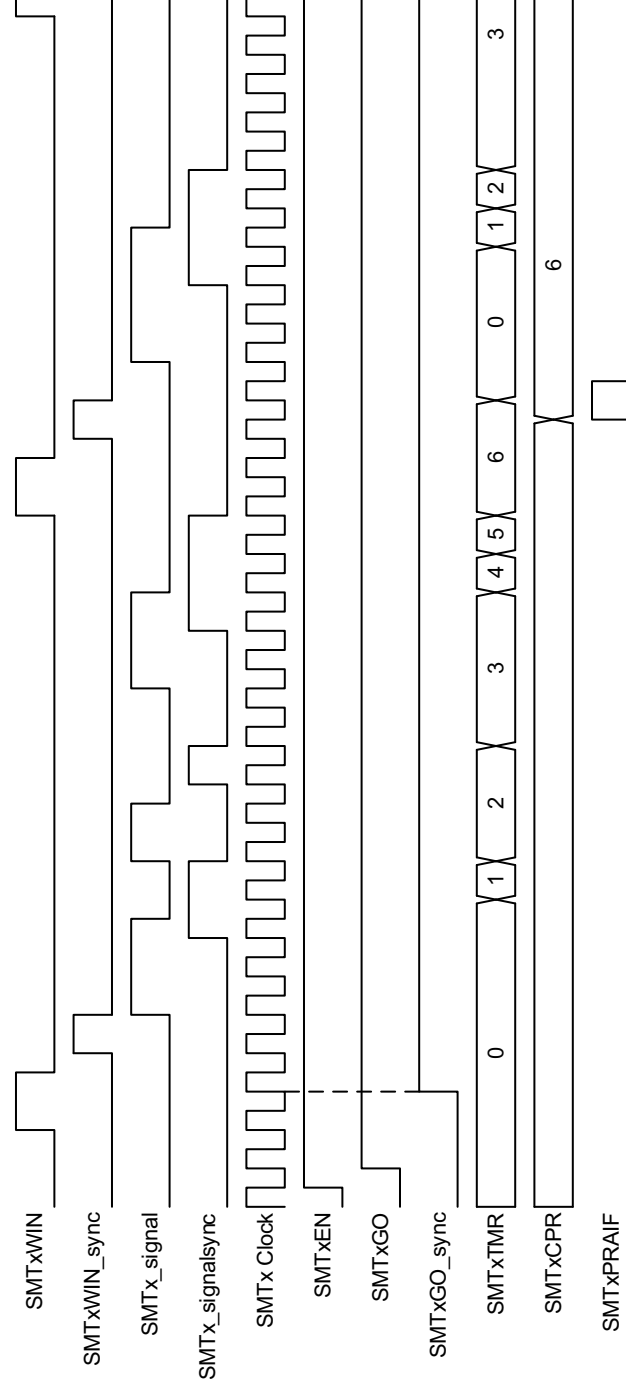
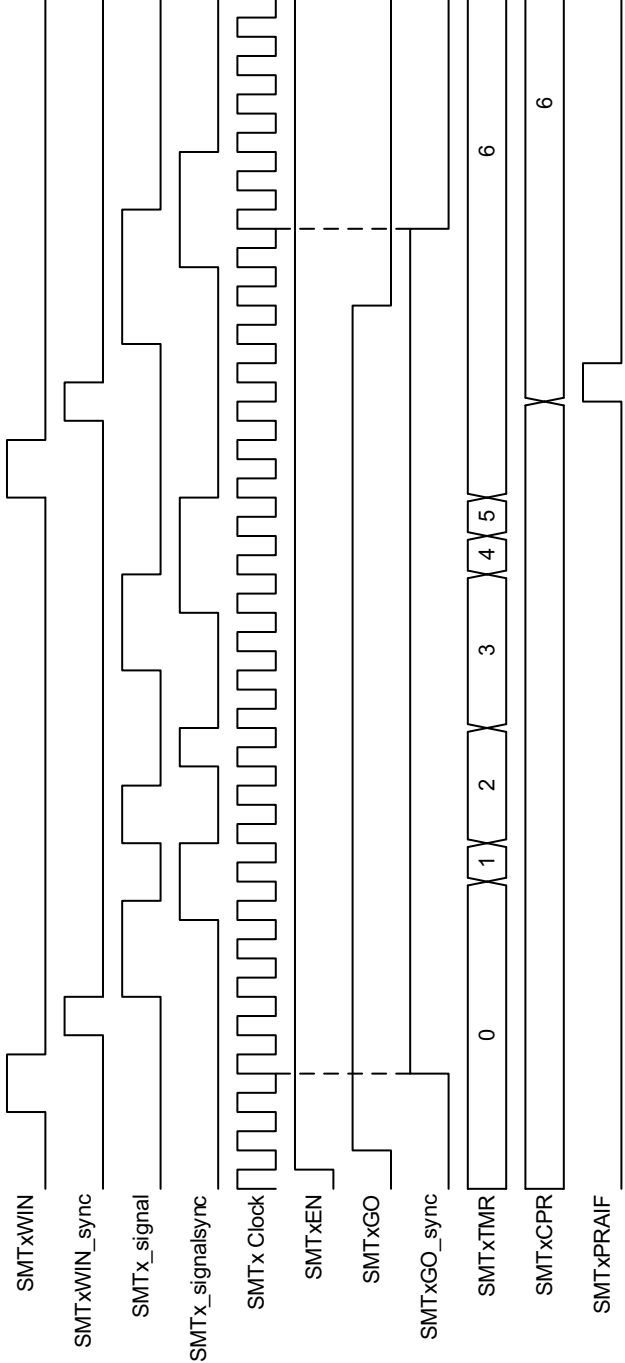


FIGURE 25-13: GATED WINDOWED MEASURE MODE SINGLE ACQUISITION TIMING DIAGRAMS



## 25.6.7 TIME OF FLIGHT MEASURE MODE

This mode measures the time interval between a rising edge on the SMTWINx input and a rising edge on the SMTx\_signal input, beginning to increment the timer upon observing a rising edge on the SMTWINx input, while updating the SMTxCPR register and resetting the timer upon observing a rising edge on the SMTx\_signal input. In the event of two SMTWINx rising edges without an SMTx\_signal rising edge, it will update the SMTxCPW register with the current value of the timer and reset the timer value. See [Figure 25-14](#) and [Figure 25-15](#).



**FIGURE 25-14: TIME OF FLIGHT MODE REPEAT ACQUISITION TIMING DIAGRAM**

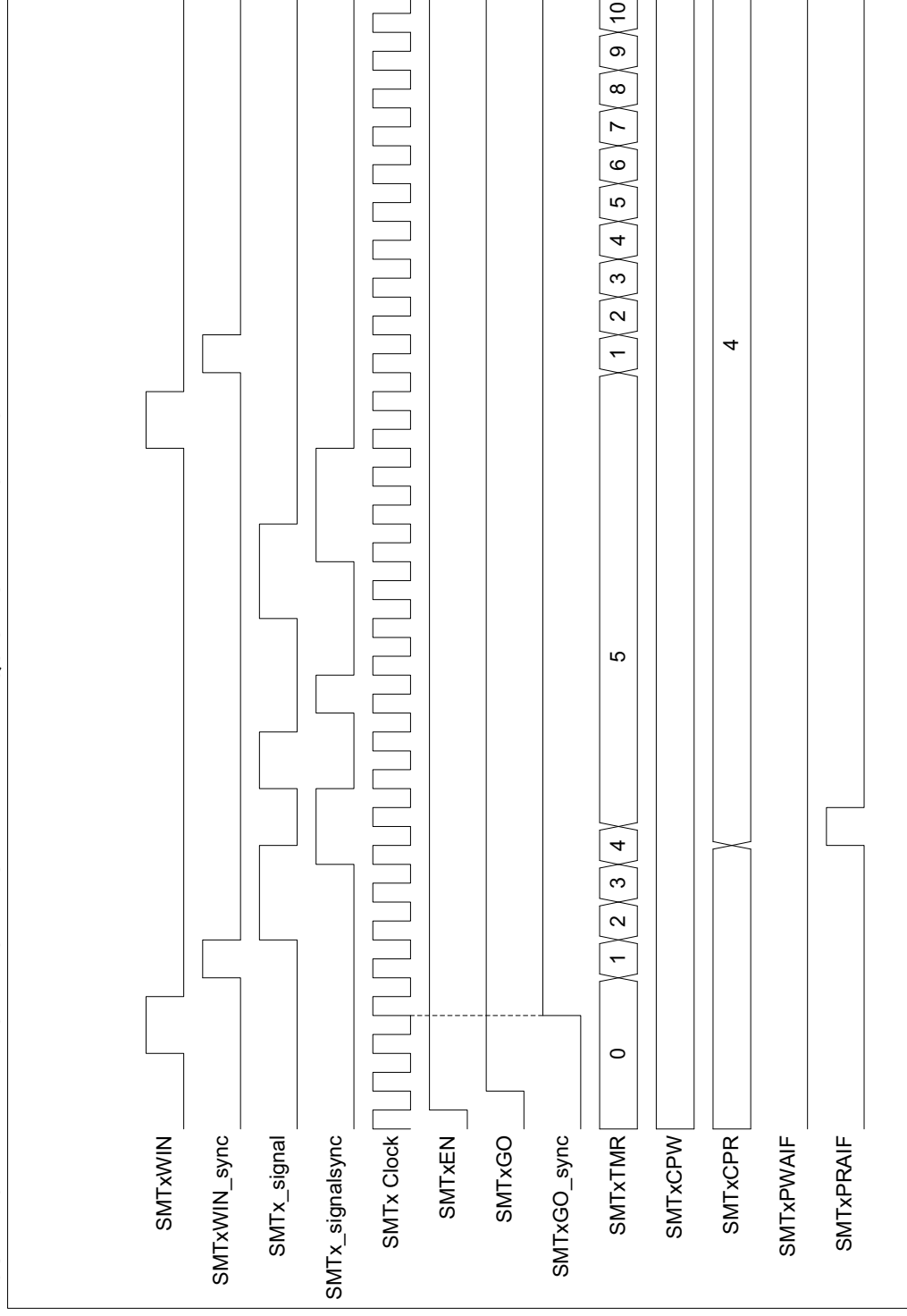
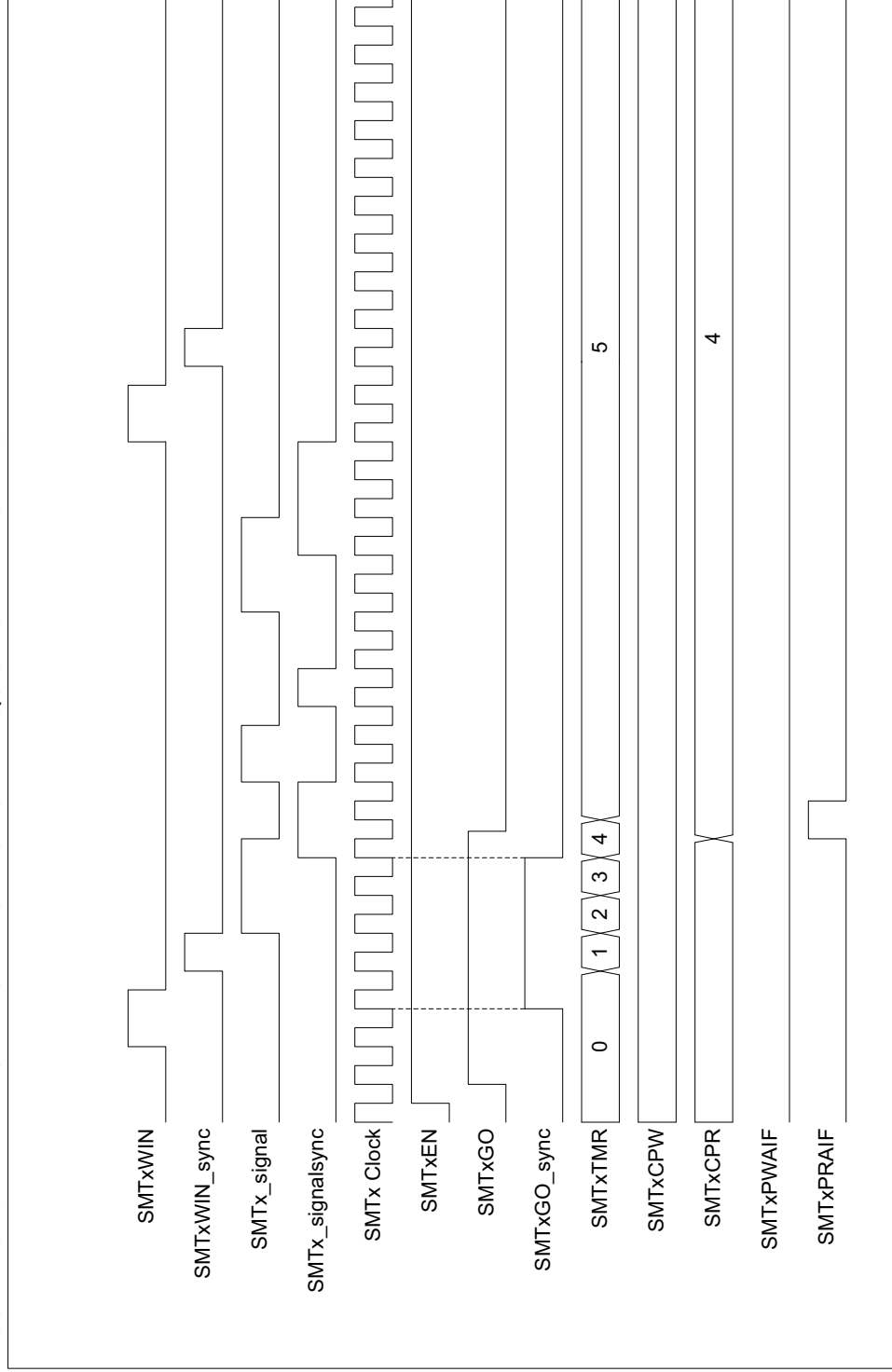


**FIGURE 25-15: TIME OF FLIGHT MODE SINGLE ACQUISITION TIMING DIAGRAM**



## 25.6.8 CAPTURE MODE

This mode captures the Timer value based on a rising or falling edge on the SMTWINx input and triggers an interrupt. This mimics the capture feature of a CCP module. The timer begins incrementing upon the GO bit being set, and updates the value of the SMTxCPR register on each rising edge of SMTWINx, and updates the value of the CPW register on each falling edge of the SMTWINx. The timer is not reset by any hardware conditions in this mode and must be reset by software, if desired. See [Figure 25-16](#) and [Figure 25-17](#).

**FIGURE 25-16: CAPTURE MODE REPEAT ACQUISITION TIMING DIAGRAM**

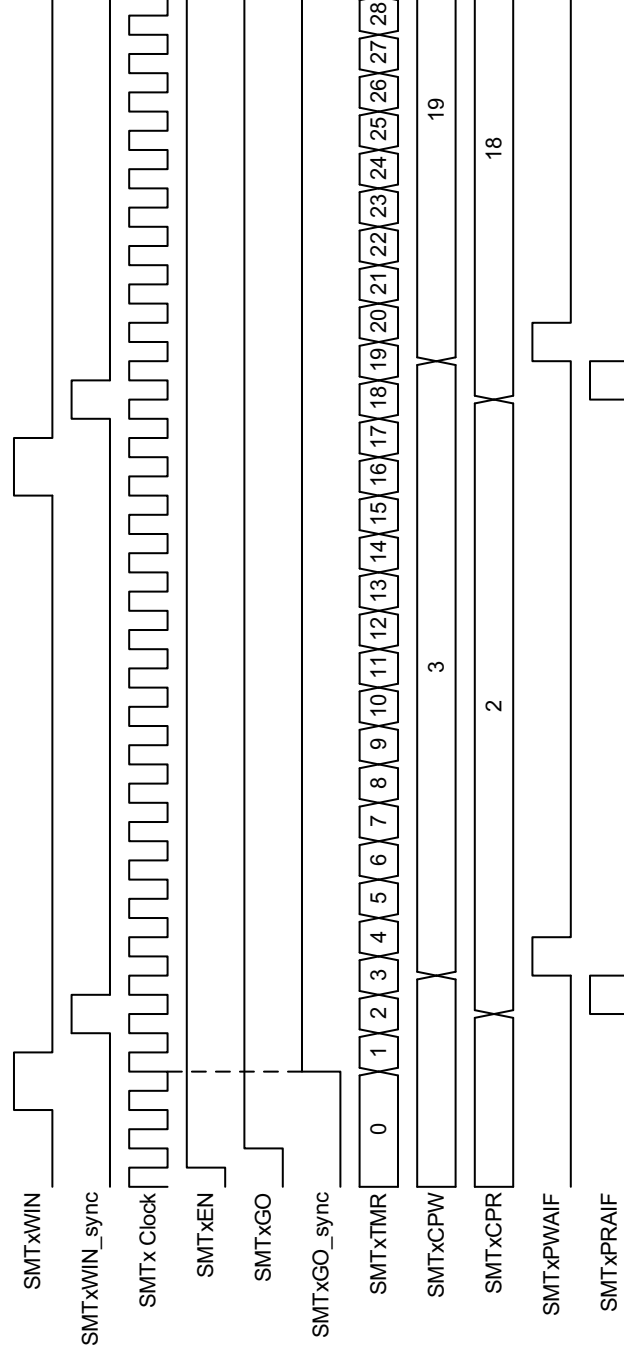
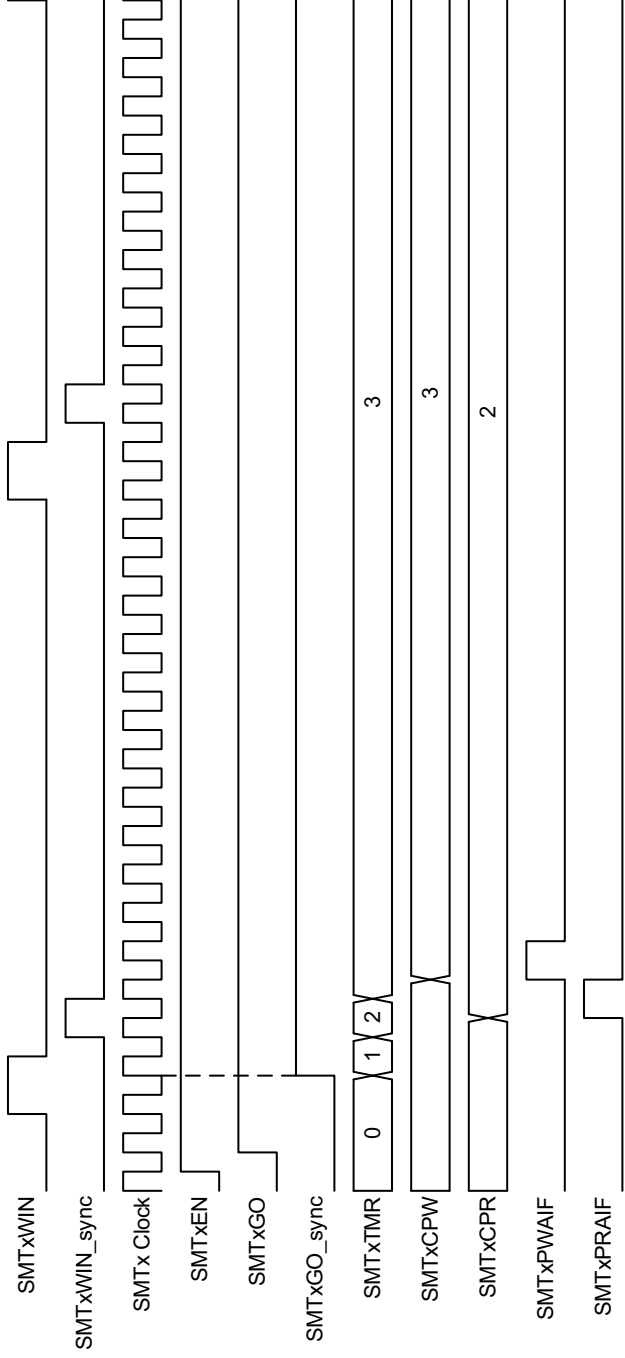


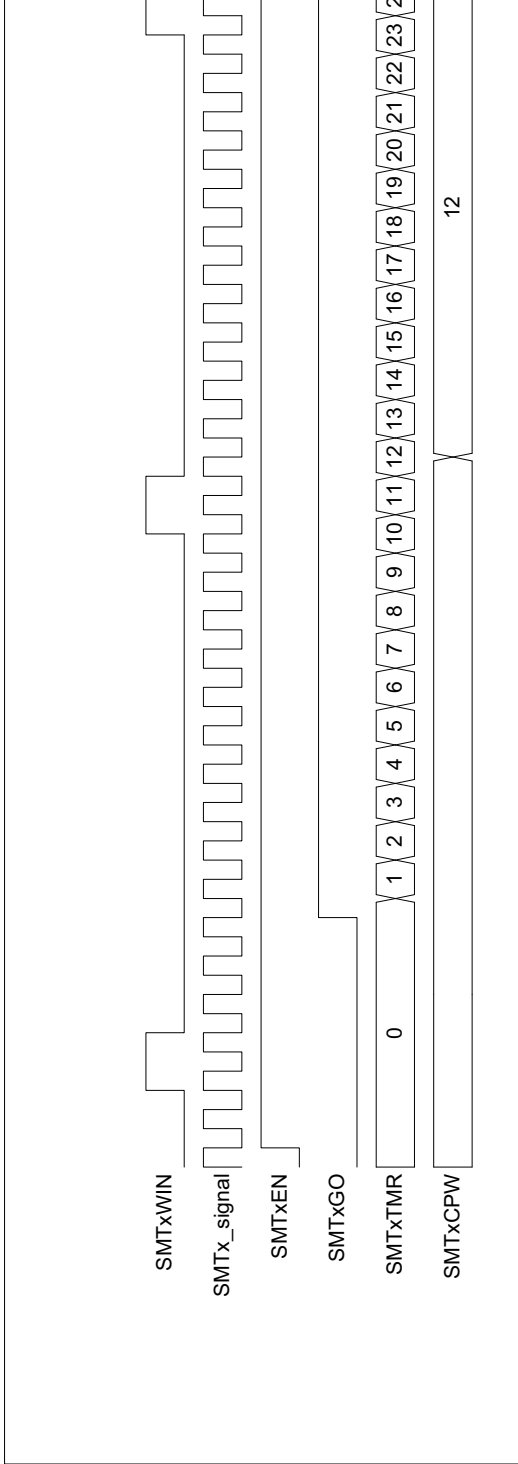
**FIGURE 25-17: CAPTURE MODE SINGLE ACQUISITION TIMING DIAGRAM**



25.6.9 COUNTER MODE

This mode increments the timer on each pulse of the SMTx\_signal input. This mode is asynchronous to the SMT clock and uses the SMTx\_signal as a time source. The SMTxCPW register will be updated with the current SMTxTMR value on the rising edge of the SMTxWIN input. See [Figure 25-18](#).

FIGURE 25-18: COUNTER MODE TIMING DIAGRAM



## 25.6.10 GATED COUNTER MODE

This mode counts pulses on the SMTx\_signal input, gated by the SMTxWIN input. It begins incrementing the timer upon seeing a rising edge of the SMTxWIN input and updates the SMTxCPW register upon a falling edge on the SMTxWIN input. See [Figure 25-19](#) and [Figure 25-20](#).

**FIGURE 25-19: GATED COUNTER MODE REPEAT ACQUISITION TIMING DIAGRAM**

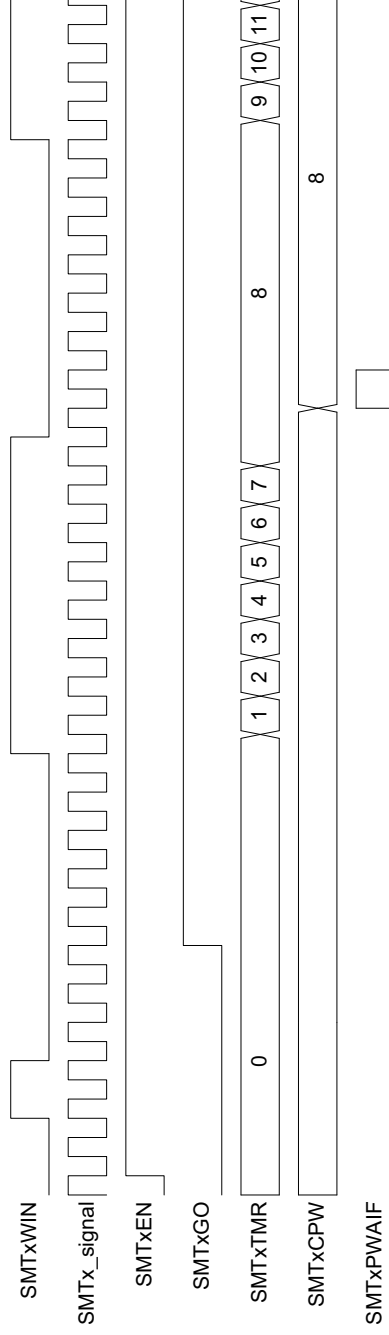
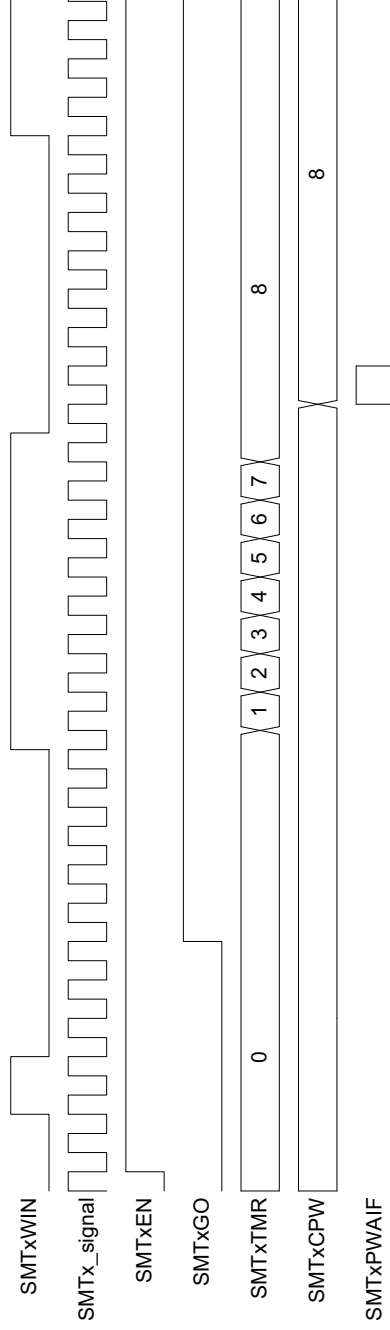


**FIGURE 25-20: GATED COUNTER MODE SINGLE ACQUISITION TIMING DIAGRAM**





## 25.6.11 WINDOWED COUNTER MODE

This mode counts pulses on the SMTx<sub>signal</sub> input, within a window dictated by the SMTxWIN input. It begins counting upon seeing a rising edge of the SMTxWIN input, updates the SMTxCPW register on a falling edge of the SMTxWIN input, and updates the SMTxCPR register on each rising edge of the SMTxWIN input beyond the first. See [Figure 25-21](#) and [Figure 25-22](#).

**FIGURE 25-21: WINDOWED COUNTER MODE REPEAT ACQUISITION TIMING DIAGRAM**

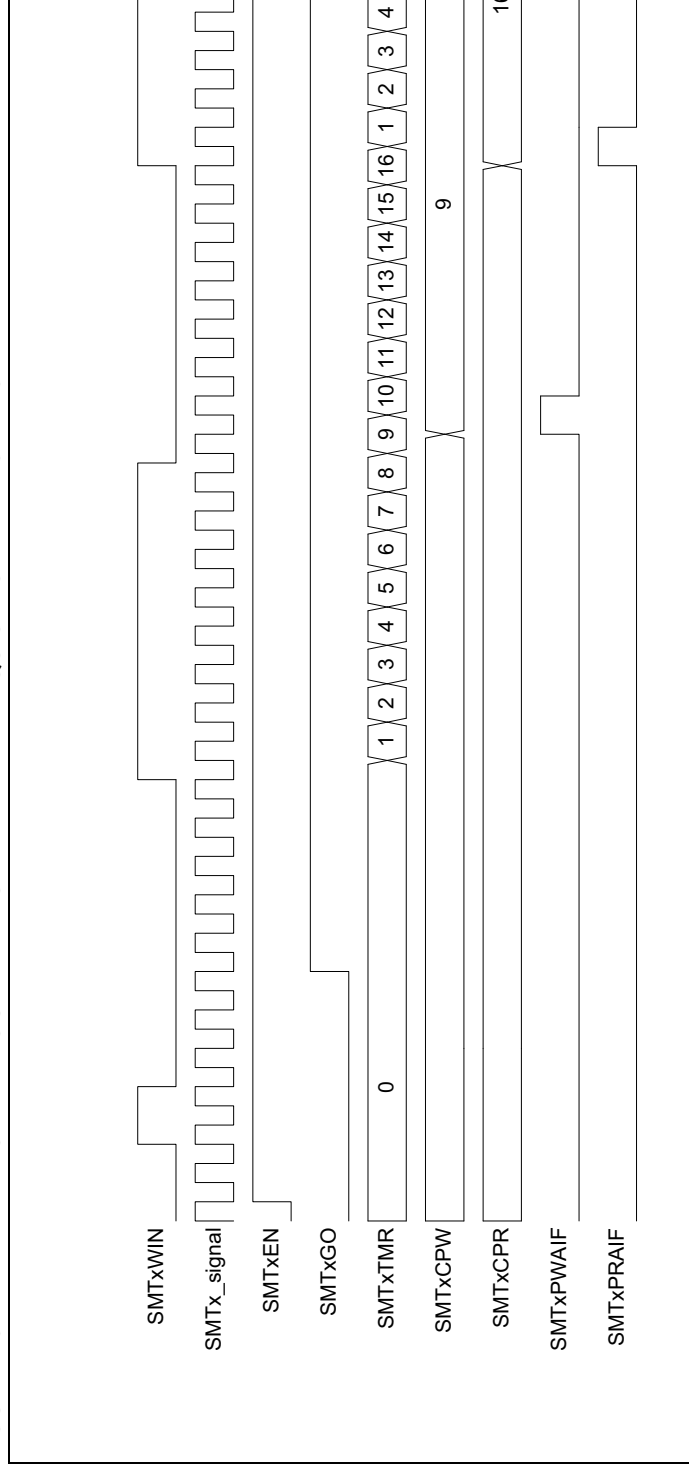
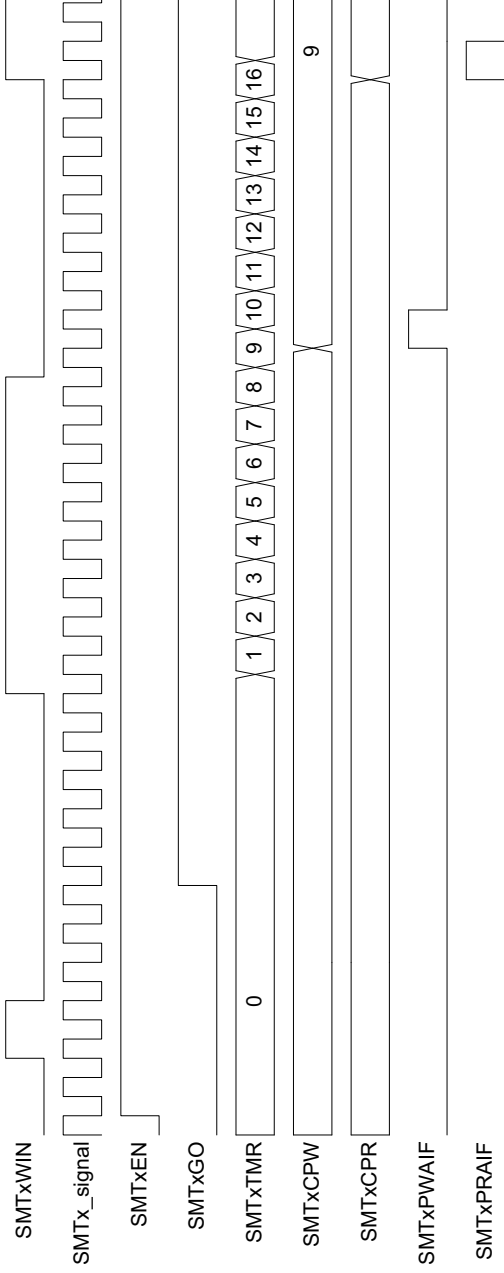


FIGURE 25-22: WINDOWED COUNTER MODE SINGLE ACQUISITION TIMING DIAGRAM



## 25.7 Interrupts

The SMT can trigger an interrupt under three different conditions:

- PW Acquisition Complete
- PR Acquisition Complete
- Counter Period Match

The interrupts are controlled by the PIR and PIE registers of the device.

### 25.7.1 PW AND PR ACQUISITION INTERRUPTS

The SMT can trigger interrupts whenever it updates the SMTxCPW and SMTxCPR registers, the circumstances for which are dependent on the SMT mode, and are discussed in each mode's specific section. The SMTxCPW interrupt is controlled by SMTxPWAIF and SMTxPWAIE bits in the respective PIR and PIE registers. The SMTxCPR interrupt is controlled by the SMTxPRAIF and SMTxPRAIE bits, also located in the respective PIR and PIE registers.

In synchronous SMT modes, the interrupt trigger is synchronized to the SMTxCLK. In Asynchronous modes, the interrupt trigger is asynchronous. In either mode, once triggered, the interrupt will be synchronized to the CPU clock.

### 25.7.2 COUNTER PERIOD MATCH INTERRUPT

As described in [Section 25.1.2 “Period Match interrupt”](#), the SMT will also interrupt upon SMTxTMR, matching SMTxPR with its period match limit functionality described in [Section 25.3 “Halt Operation”](#). The period match interrupt is controlled by SMTxIF and SMTxIE, located in the respective PIR and PIE registers.

## 25.8 Register Definitions: SMT Control

Long bit name prefixes for the Signal Measurement Timer peripherals are shown in [Section 1.3 “Register and Bit naming conventions”](#).

**TABLE 25-2: LONG BIT NAMES PREFIXES FOR SMT PERIPHERALS**

| Peripheral | Bit Name Prefix |
|------------|-----------------|
| SMT1       | SMT1            |
| SMT2       | SMT2            |

**REGISTER 25-1: SMTxCON0: SMT CONTROL REGISTER 0**

| R/W-0/0           | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|-------------------|-----|---------|---------|---------|---------|---------|---------|
| EN <sup>(1)</sup> | —   | STP     | WPOL    | SPOL    | CPOL    | PS<1:0> |         |
| bit 7             |     |         |         |         |         |         | bit 0   |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as ‘0’

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

‘1’ = Bit is set

‘0’ = Bit is cleared

- bit 7 **EN:** SMT Enable bit<sup>(1)</sup>  
 1 = SMT is enabled  
 0 = SMT is disabled; internal states are reset, clock requests are disabled
- bit 6 **Unimplemented:** Read as ‘0’
- bit 5 **STP:** SMT Counter Halt Enable bit  
 When SMTxTMR = SMTxPR:  
 1 = Counter remains SMTxPR; period match interrupt occurs when clocked  
 0 = Counter resets to 24'h000000; period match interrupt occurs when clocked
- bit 4 **WPOL:** SMTxWIN Input Polarity Control bit  
 1 = SMTxWIN signal is active-low/falling edge enabled  
 0 = SMTxWIN signal is active-high/rising edge enabled
- bit 3 **SPOL:** SMTxSIG Input Polarity Control bit  
 1 = SMTx\_signal is active-low/falling edge enabled  
 0 = SMTx\_signal is active-high/rising edge enabled
- bit 2 **CPOL:** SMT Clock Input Polarity Control bit  
 1 = SMTxTMR increments on the falling edge of the selected clock signal  
 0 = SMTxTMR increments on the rising edge of the selected clock signal
- bit 1-0 **PS<1:0>:** SMT Prescale Select bits  
 11 = Prescaler = 1:8  
 10 = Prescaler = 1:4  
 01 = Prescaler = 1:2  
 00 = Prescaler = 1:1

**Note 1:** Setting EN to ‘0’ does not affect the register contents.

## REGISTER 25-2: SMTxCON1: SMT CONTROL REGISTER 1

| R/W/HC-0/0 | R/W-0/0 | U-0 | U-0 | R/W-0/0   | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|------------|---------|-----|-----|-----------|---------|---------|---------|
| GO         | REPEAT  | —   | —   | MODE<3:0> |         |         |         |
| bit 7      |         |     |     | bit 0     |         |         |         |

### Legend:

HC = Bit is cleared by hardware

HS = Bit is set by hardware

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

q = Value depends on condition

- bit 7      **GO:** GO Data Acquisition bit  
1 = Incrementing, acquiring data is enabled  
0 = Incrementing, acquiring data is disabled
- bit 6      **REPEAT:** SMT Repeat Acquisition Enable bit  
1 = Repeat Data Acquisition mode is enabled  
0 = Single Acquisition mode is enabled
- bit 5-4    **Unimplemented:** Read as '0'
- bit 3-0    **MODE<3:0>** SMT Operation Mode Select bits  
1111 = Reserved  
•  
•  
•  
1011 = Reserved  
1010 = Windowed counter  
1001 = Gated counter  
1000 = Counter  
0111 = Capture  
0110 = Time of flight  
0101 = Gated windowed measure  
0100 = Windowed measure  
0011 = High and low time measurement  
0010 = Period and Duty-Cycle Acquisition  
0001 = Gated Timer  
0000 = Timer

## REGISTER 25-3: SMTxSTAT: SMT STATUS REGISTER

|            |            |            |     |     |       |       |       |
|------------|------------|------------|-----|-----|-------|-------|-------|
| R/W/HC-0/0 | R/W/HC-0/0 | R/W/HC-0/0 | U-0 | U-0 | R-0/0 | R-0/0 | R-0/0 |
| CPRUP      | CPWUP      | RST        | —   | —   | TS    | WS    | AS    |
| bit 7      |            |            |     |     |       |       | bit 0 |

### Legend:

HC = Bit is cleared by hardware

HS = Bit is set by hardware

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

q = Value depends on condition

- bit 7      **CPRUP:** SMT Manual Period Buffer Update bit  
1 = Request update to SMTxCPRx registers  
0 = SMTxCPRx registers update is complete
- bit 6      **CPWUP:** SMT Manual Pulse Width Buffer Update bit  
1 = Request update to SMTxCPW registers  
0 = SMTxCPW registers update is complete
- bit 5      **RST:** SMT Manual Timer Reset bit  
1 = Request Reset to SMTxTMR registers  
0 = SMTxTMR registers update is complete
- bit 4-3    **Unimplemented:** Read as '0'
- bit 2      **TS:** GO Value Status bit  
1 = SMT timer is incrementing  
0 = SMT timer is not incrementing
- bit 1      **WS:** SMTxWIN Value Status bit  
1 = SMT window is open  
0 = SMT window is closed
- bit 0      **AS:** SMT\_signal Value Status bit  
1 = SMT acquisition is in progress  
0 = SMT acquisition is not in progress

**REGISTER 25-4: SMTxCLK: SMT CLOCK SELECTION REGISTER**

|       |     |     |     |     |           |         |         |
|-------|-----|-----|-----|-----|-----------|---------|---------|
| U-0   | U-0 | U-0 | U-0 | U-0 | R/W-0/0   | R/W-0/0 | R/W-0/0 |
| —     | —   | —   | —   | —   | CSEL<2:0> |         |         |
| bit 7 |     |     |     |     | bit 0     |         |         |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

q = Value depends on condition

bit 7-3

**Unimplemented:** Read as '0'

bit 2-0

**CSEL<2:0>:** SMT Clock Selection bits

111 = Reference Clock Output

110 = SOSC

101 = MFINTOSC/16 (32 kHz)

100 = MFINTOSC (500 kHz)

011 = LFINTOSC

010 = HFINTOSC 16 MHz

001 = Fosc

000 = Fosc/4



## REGISTER 25-5: SMTxWIN: SMTx WINDOW INPUT SELECT REGISTER

| U-0   | U-0 | U-0 | R/W-0/0   | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|-------|-----|-----|-----------|---------|---------|---------|---------|
| —     | —   | —   | WSEL<4:0> |         |         |         |         |
| bit 7 |     |     |           |         |         |         |         |
|       |     |     |           |         |         |         |         |
|       |     |     |           |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

q = Value depends on condition

bit 7-5

**Unimplemented:** Read as '0'

bit 4-0

**WSEL<4:0>:** SMTx Window Selection bits

11111 = Reserved

.

.

.

11011 = Reserved

11010 = CLC4\_out

11001 = CLC3\_out

11000 = CLC2\_out

10111 = CLC1\_out

10110 = ZCD1\_out

10101 = CMP2\_out

10100 = CMP1\_out

10011 = NCO1\_out

10010 = Reserved

10001 = Reserved

10000 = PWM8\_out

01111 = PWM7\_out

01110 = PWM6\_out

01101 = PWM5\_out

01100 = CCP4\_out

01011 = CCP3\_out

01010 = CCP2\_out

01001 = CCP1\_out

01000 = TMR6\_postscaled

00111 = TMR4\_postscaled

00110 = TMR2\_postscaled

00101 = TMR0\_overflow

00100 = CLKREF

00011 = SOSC

00010 = MFINTOSC/16 (32 kHz)

00001 = LFINTOSC

00000 = SMTxWINPPS

## REGISTER 25-6: SMTxSIG: SMTx SIGNAL INPUT SELECT REGISTER

| U-0   | U-0 | U-0 | R/W-0/0   | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|-------|-----|-----|-----------|---------|---------|---------|---------|
| —     | —   | —   | SSEL<4:0> |         |         |         |         |
| bit 7 |     |     |           |         |         |         |         |
|       |     |     |           |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

q = Value depends on condition

bit 7-5

**Unimplemented:** Read as '0'

bit 4-0

**SSEL<4:0>:** SMTx Signal Selection bits

11111 = Reserved

•  
•  
•

11010 = CAN\_rx\_timestamp

11001 = CLC4\_out

11000 = CLC3\_out

10111 = CLC2\_out

10110 = CLC1\_out

10101 = ZCD1\_out

10100 = CMP2\_out

10011 = CMP1\_out

10010 = NCO1\_out

10001 = Reserved

10000 = Reserved

01111 = PWM8\_out

01110 = PWM7\_out

01101 = PWM6\_out

01100 = PWM5\_out

01011 = CCP4\_out

01010 = CCP3\_out

01001 = CCP2\_out

01000 = CCP1\_out

00111 = TMR6\_postscaled

00110 = TMR5\_overflow

00101 = TMR4\_postscaled

00100 = TMR3\_overflow

00011 = TMR2\_postscaled

00010 = TMR1\_overflow

00001 = TMR0\_overflow

00000 = SMTxSIGPPS

## REGISTER 25-7: SMTxTMRL: SMT TIMER REGISTER – LOW BYTE

|              |         |         |         |         |         |         |         |
|--------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0      | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| SMTxTMR<7:0> |         |         |         |         |         |         |         |
| bit 7        |         |         |         | bit 0   |         |         |         |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-0

**SMTxTMR<7:0>**: Significant bits of the SMT Counter – Low Byte

## REGISTER 25-8: SMTxTMRH: SMT TIMER REGISTER – HIGH BYTE

|               |         |         |         |         |         |         |         |
|---------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0       | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| SMTxTMR<15:8> |         |         |         |         |         |         |         |
| bit 7         |         |         |         | bit 0   |         |         |         |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-0

**SMTxTMR<15:8>**: Significant bits of the SMT Counter – High Byte

## REGISTER 25-9: SMTxTMRU: SMT TIMER REGISTER – UPPER BYTE

|                |         |         |         |         |         |         |         |
|----------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0        | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| SMTxTMR<23:16> |         |         |         |         |         |         |         |
| bit 7          |         |         |         | bit 0   |         |         |         |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-0

**SMTxTMR<23:16>**: Significant bits of the SMT Counter – Upper Byte

## REGISTER 25-10: SMTxCPR<sub>L</sub>: SMT CAPTURED PERIOD REGISTER – LOW BYTE

|              |       |       |       |       |       |       |       |
|--------------|-------|-------|-------|-------|-------|-------|-------|
| R-x/x        | R-x/x | R-x/x | R-x/x | R-x/x | R-x/x | R-x/x | R-x/x |
| SMTxCPR<7:0> |       |       |       |       |       |       |       |
| bit 7        |       |       |       | bit 0 |       |       |       |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0      **SMTxCPR<7:0>**: Significant bits of the SMT Period Latch – Low Byte

## REGISTER 25-11: SMTxCPR<sub>H</sub>: SMT CAPTURED PERIOD REGISTER – HIGH BYTE

|               |       |       |       |       |       |       |       |
|---------------|-------|-------|-------|-------|-------|-------|-------|
| R-x/x         | R-x/x | R-x/x | R-x/x | R-x/x | R-x/x | R-x/x | R-x/x |
| SMTxCPR<15:8> |       |       |       |       |       |       |       |
| bit 7         |       |       |       | bit 0 |       |       |       |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0      **SMTxCPR<15:8>**: Significant bits of the SMT Period Latch – High Byte

## REGISTER 25-12: SMTxCPR<sub>U</sub>: SMT CAPTURED PERIOD REGISTER – UPPER BYTE

|                |       |       |       |       |       |       |       |
|----------------|-------|-------|-------|-------|-------|-------|-------|
| R-x/x          | R-x/x | R-x/x | R-x/x | R-x/x | R-x/x | R-x/x | R-x/x |
| SMTxCPR<23:16> |       |       |       |       |       |       |       |
| bit 7          |       |       |       | bit 0 |       |       |       |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0      **SMTxCPR<23:16>**: Significant bits of the SMT Period Latch – Upper Byte

## REGISTER 25-13: SMTxCPWL: SMT CAPTURED PULSE WIDTH REGISTER – LOW BYTE

|              |       |       |       |       |       |       |       |
|--------------|-------|-------|-------|-------|-------|-------|-------|
| R-x/x        | R-x/x | R-x/x | R-x/x | R-x/x | R-x/x | R-x/x | R-x/x |
| SMTxCPW<7:0> |       |       |       |       |       |       |       |
| bit 7        |       |       |       | bit 0 |       |       |       |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0 **SMTxCPW<7:0>**: Significant bits of the SMT PW Latch – Low Byte

## REGISTER 25-14: SMTxCPWH: SMT CAPTURED PULSE WIDTH REGISTER – HIGH BYTE

|               |       |       |       |       |       |       |       |
|---------------|-------|-------|-------|-------|-------|-------|-------|
| R-x/x         | R-x/x | R-x/x | R-x/x | R-x/x | R-x/x | R-x/x | R-x/x |
| SMTxCPW<15:8> |       |       |       |       |       |       |       |
| bit 7         |       |       |       | bit 0 |       |       |       |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0 **SMTxCPW<15:8>**: Significant bits of the SMT PW Latch – High Byte

## REGISTER 25-15: SMTxCPWU: SMT CAPTURED PULSE WIDTH REGISTER – UPPER BYTE

|                |       |       |       |       |       |       |       |
|----------------|-------|-------|-------|-------|-------|-------|-------|
| R-x/x          | R-x/x | R-x/x | R-x/x | R-x/x | R-x/x | R-x/x | R-x/x |
| SMTxCPW<23:16> |       |       |       |       |       |       |       |
| bit 7          |       |       |       | bit 0 |       |       |       |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0 **SMTxCPW<23:16>**: Significant bits of the SMT PW Latch – Upper Byte

## REGISTER 25-16: SMTxPRL: SMT PERIOD REGISTER – LOW BYTE

|             |         |         |         |         |         |         |         |
|-------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-x/1     | R/W-x/1 | R/W-x/1 | R/W-x/1 | R/W-x/1 | R/W-x/1 | R/W-x/1 | R/W-x/1 |
| SMTxPR<7:0> |         |         |         |         |         |         |         |
| bit 7       |         |         |         | bit 0   |         |         |         |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0 **SMTxPR<7:0>**: Significant bits of the SMT Timer Value for Period Match – Low Byte

## REGISTER 25-17: SMTxPRH: SMT PERIOD REGISTER – HIGH BYTE

|              |         |         |         |         |         |         |         |
|--------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-x/1      | R/W-x/1 | R/W-x/1 | R/W-x/1 | R/W-x/1 | R/W-x/1 | R/W-x/1 | R/W-x/1 |
| SMTxPR<15:8> |         |         |         |         |         |         |         |
| bit 7        |         |         |         | bit 0   |         |         |         |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0 **SMTxPR<15:8>**: Significant bits of the SMT Timer Value for Period Match – High Byte

## REGISTER 25-18: SMTxPRU: SMT PERIOD REGISTER – UPPER BYTE

|               |         |         |         |         |         |         |         |
|---------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-x/1       | R/W-x/1 | R/W-x/1 | R/W-x/1 | R/W-x/1 | R/W-x/1 | R/W-x/1 | R/W-x/1 |
| SMTxPR<23:16> |         |         |         |         |         |         |         |
| bit 7         |         |         |         | bit 0   |         |         |         |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0 **SMTxPR<23:16>**: Significant bits of the SMT Timer Value for Period Match – Upper Byte

**TABLE 25-3: SUMMARY OF REGISTERS ASSOCIATED WITH SMTx**

| Name     | Bit 7      | Bit 6  | Bit 5 | Bit 4     | Bit 3     | Bit 2     | Bit 1       | Bit 0 | Register on Page |
|----------|------------|--------|-------|-----------|-----------|-----------|-------------|-------|------------------|
| SMT1CON0 | EN         | —      | STP   | WPOL      | SPOL      | CPOL      | SMT1PS<1:0> |       | 381              |
| SMT1CON1 | GO         | REPEAT | —     | —         | MODE<3:0> |           |             |       | 382              |
| SMT1STAT | CPRUP      | CPWUP  | RST   | —         | —         | TS        | WS          | AS    | 383              |
| SMT1CLK  | —          | —      | —     | —         | —         | CSEL<2:0> |             |       | 384              |
| SMT1SIG  | —          | —      | —     | SSEL<4:0> |           |           |             |       | 386              |
| SMT1WIN  | —          | —      | —     | WSEL<4:0> |           |           |             |       | 385              |
| SMT1TMRL | TMR<7:0>   |        |       |           |           |           |             |       | 387              |
| SMT1TMRH | TMR<15:8>  |        |       |           |           |           |             |       | 387              |
| SMT1TMRU | TMR<23:16> |        |       |           |           |           |             |       | 387              |
| SMT1CPRL | CPR<7:0>   |        |       |           |           |           |             |       | 388              |
| SMT1CPRH | CPR<15:8>  |        |       |           |           |           |             |       | 388              |
| SMT1CPRU | CPR<23:16> |        |       |           |           |           |             |       | 388              |
| SMT1CPWL | CPW<7:0>   |        |       |           |           |           |             |       | 389              |
| SMT1CPWH | CPW<15:8>  |        |       |           |           |           |             |       | 389              |
| SMT1CPWU | CPW<23:16> |        |       |           |           |           |             |       | 389              |
| SMT1PRL  | PR<7:0>    |        |       |           |           |           |             |       | 390              |
| SMT1PRH  | PR<15:8>   |        |       |           |           |           |             |       | 390              |
| SMT1PRU  | PR<23:16>  |        |       |           |           |           |             |       | 390              |
| SMT2CON0 | EN         | —      | STP   | WPOL      | SPOL      | CPOL      | SMT2PS<1:0> |       | 381              |
| SMT2CON1 | GO         | REPEAT | —     | —         | MODE<3:0> |           |             |       | 382              |
| SMT2STAT | CPRUP      | CPWUP  | RST   | —         | —         | TS        | WS          | AS    | 383              |
| SMT2CLK  | —          | —      | —     | —         | —         | CSEL<2:0> |             |       | 384              |
| SMT2SIG  | —          | —      | —     | SSEL<4:0> |           |           |             |       | 386              |
| SMT2WIN  | —          | —      | —     | WSEL<4:0> |           |           |             |       | 385              |
| SMT2TMRL | TMR<7:0>   |        |       |           |           |           |             |       | 387              |
| SMT2TMRH | TMR<15:8>  |        |       |           |           |           |             |       | 387              |
| SMT2TMRU | TMR<23:16> |        |       |           |           |           |             |       | 387              |
| SMT2CPRL | CPR<7:0>   |        |       |           |           |           |             |       | 388              |
| SMT2CPRH | CPR<15:8>  |        |       |           |           |           |             |       | 388              |
| SMT2CPRU | CPR<23:16> |        |       |           |           |           |             |       | 388              |
| SMT2CPWL | CPW<7:0>   |        |       |           |           |           |             |       | 389              |
| SMT2CPWH | CPW<15:8>  |        |       |           |           |           |             |       | 389              |
| SMT2CPWU | CPW<23:16> |        |       |           |           |           |             |       | 389              |
| SMT2PRL  | PR<7:0>    |        |       |           |           |           |             |       | 390              |
| SMT2PRH  | PR<15:8>   |        |       |           |           |           |             |       | 390              |
| SMT2PRU  | PR<23:16>  |        |       |           |           |           |             |       | 390              |

**Legend:** — = unimplemented read as '0'. Shaded cells are not used for the SMTx module.

## 26.0 COMPLEMENTARY WAVEFORM GENERATOR (CWG) MODULE

The Complementary Waveform Generator (CWG) produces half-bridge, full-bridge, and steering of PWM waveforms. It is backwards compatible with previous CCP functions. The PIC18(L)F25/26K83 family has three instances of the CWG module.

Each of the CWG modules has the following features:

- Six operating modes:
  - Synchronous Steering mode
  - Asynchronous Steering mode
  - Full-Bridge mode, Forward
  - Full-Bridge mode, Reverse
  - Half-Bridge mode
  - Push-Pull mode
- Output polarity control
- Output steering
- Independent 6-bit rising and falling event dead-band timers
  - Clocked dead band
  - Independent rising and falling dead-band enables
- Auto-shutdown control with:
  - Selectable shutdown sources
  - Auto-restart option
  - Auto-shutdown pin override control

### 26.1 Fundamental Operation

The CWG generates two output waveforms from the selected input source.

The off-to-on transition of each output can be delayed from the on-to-off transition of the other output, thereby, creating a time delay immediately where neither output is driven. This is referred to as dead time and is covered in [Section 26.6 “Dead-Band Control”](#).

It may be necessary to guard against the possibility of circuit faults or a feedback event arriving too late or not at all. In this case, the active drive must be terminated before the Fault condition causes damage. This is referred to as auto-shutdown and is covered in [Section 26.10 “Auto-Shutdown”](#).

## 26.2 Operating Modes

The CWG module can operate in six different modes, as specified by the MODE<2:0> bits of the CWGxCON0 register:

- Half-Bridge mode
- Push-Pull mode
- Asynchronous Steering mode
- Synchronous Steering mode
- Full-Bridge mode, Forward
- Full-Bridge mode, Reverse

All modes accept a single pulse data input, and provide up to four outputs as described in the following sections.

All modes include auto-shutdown control as described in [Section 26.10 “Auto-Shutdown”](#).

**Note:** Except as noted for Full-bridge mode ([Section 26.2.3 “Full-Bridge Modes”](#)), mode changes should only be performed while EN = 0 ([Register 26-1](#)).

### 26.2.1 HALF-BRIDGE MODE

In Half-Bridge mode, two output signals are generated as true and inverted versions of the input as illustrated in [Figure 26-2](#). A non-overlap (dead-band) time is inserted between the two outputs as described in [Section 26.6 “Dead-Band Control”](#). The output steering feature cannot be used in this mode. A basic block diagram of this mode is shown in [Figure 26-1](#).

The unused outputs CWGxC and CWGxD drive similar signals as CWGxA and CWGxB, with polarity independently controlled by the POLC and POLD bits of the CWGxCON1 register, respectively.



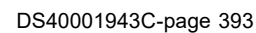
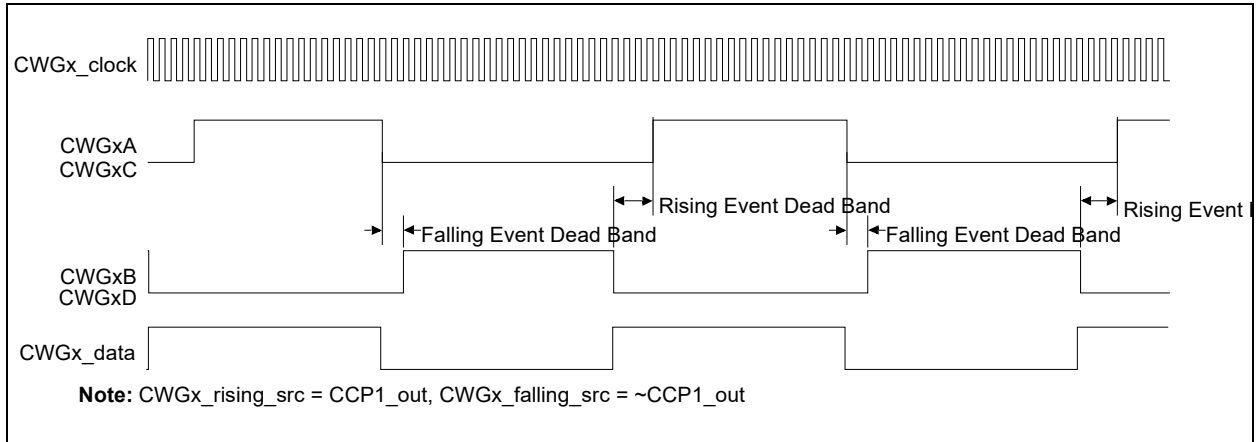


**FIGURE 26-2: CWGx HALF-BRIDGE MODE OPERATION**



## 26.2.2 PUSH-PULL MODE

In Push-Pull mode, two output signals are generated, alternating copies of the input as illustrated in [Figure 26-4](#). This alternation creates the push-pull effect required for driving some transformer-based power supply designs. Steering modes are not used in Push-Pull mode. A basic block diagram for the Push-Pull mode is shown in [Figure 26-3](#).

The push-pull sequencer is reset whenever EN = 0 or if an auto-shutdown event occurs. The sequencer is clocked by the first input pulse, and the first output appears on CWGxA.

The unused outputs CWGxC and CWGxD drive copies of CWGxA and CWGxB, respectively, but with polarity controlled by the POLC and POLD bits of the CWGxCON1 register, respectively.

**FIGURE 26-3: SIMPLIFIED CWG BLOCK DIAGRAM (PUSH-PULL MODE, MODE<2:0> = 101)**

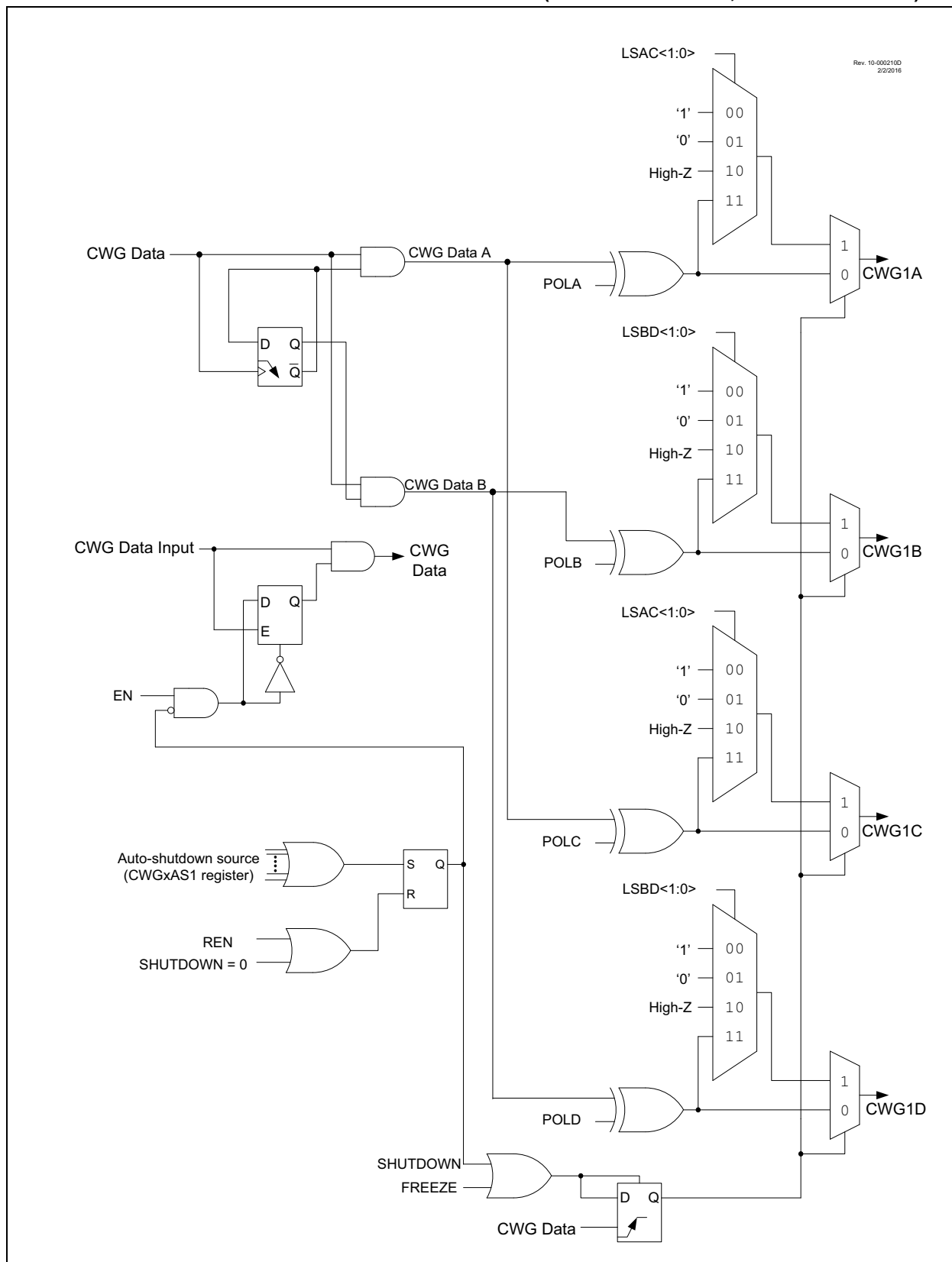
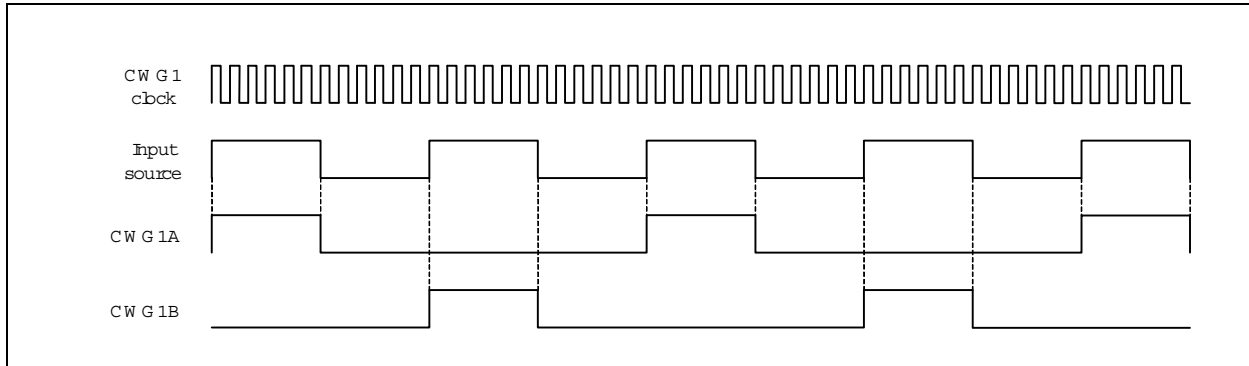


**FIGURE 26-4: CWGx PUSH-PULL MODE OPERATION**



## 26.2.3 FULL-BRIDGE MODES

In Forward and Reverse Full-Bridge modes, three outputs drive static values while the fourth is modulated by the input data signal. The mode selection may be toggled between forward and reverse by toggling the  $MODE<0>$  bit of the  $CWGxCON0$  while keeping  $MODE<2:1>$  static, without disabling the CWG module. When connected as shown in [Figure 26-5](#), the outputs are appropriate for a full-bridge motor driver. Each CWG output signal has independent polarity control, so the circuit can be adapted to high-active and low-active drivers. A simplified block diagram for the Full-Bridge modes is shown in [Figure 26-6](#).

**FIGURE 26-5: EXAMPLE OF FULL-BRIDGE APPLICATION**

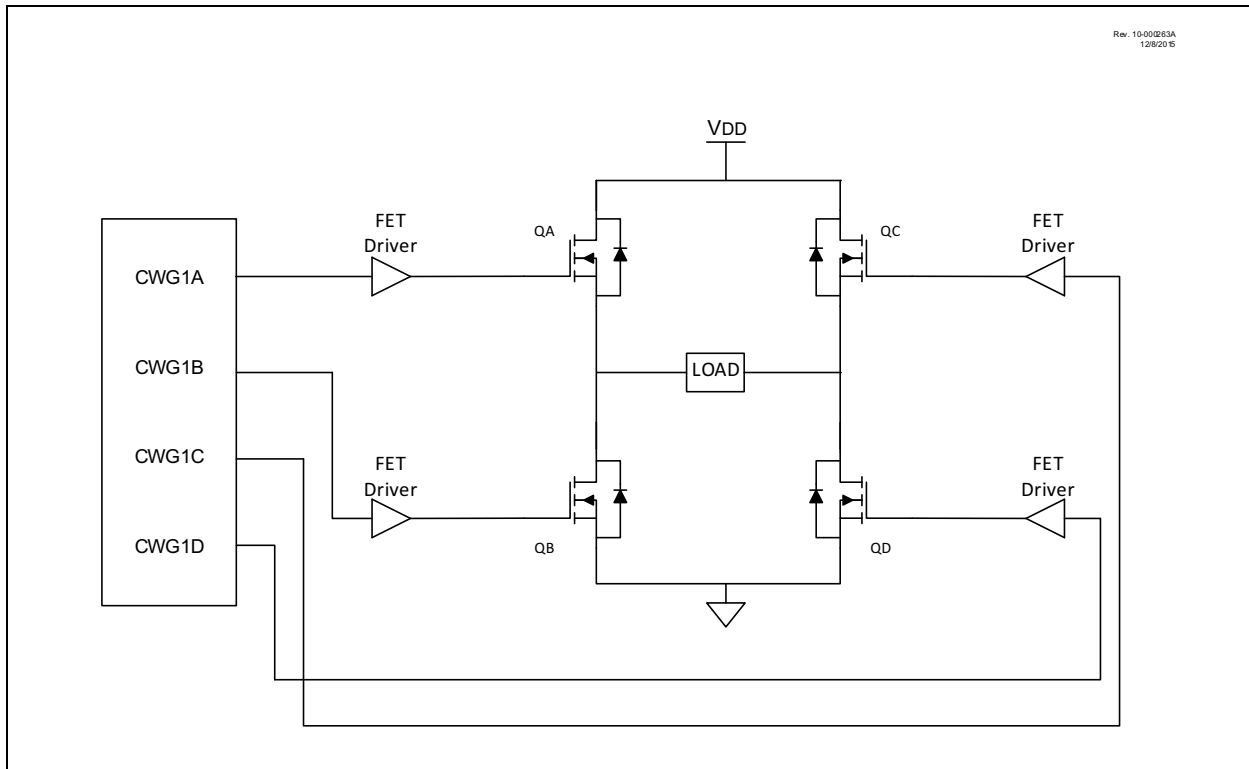
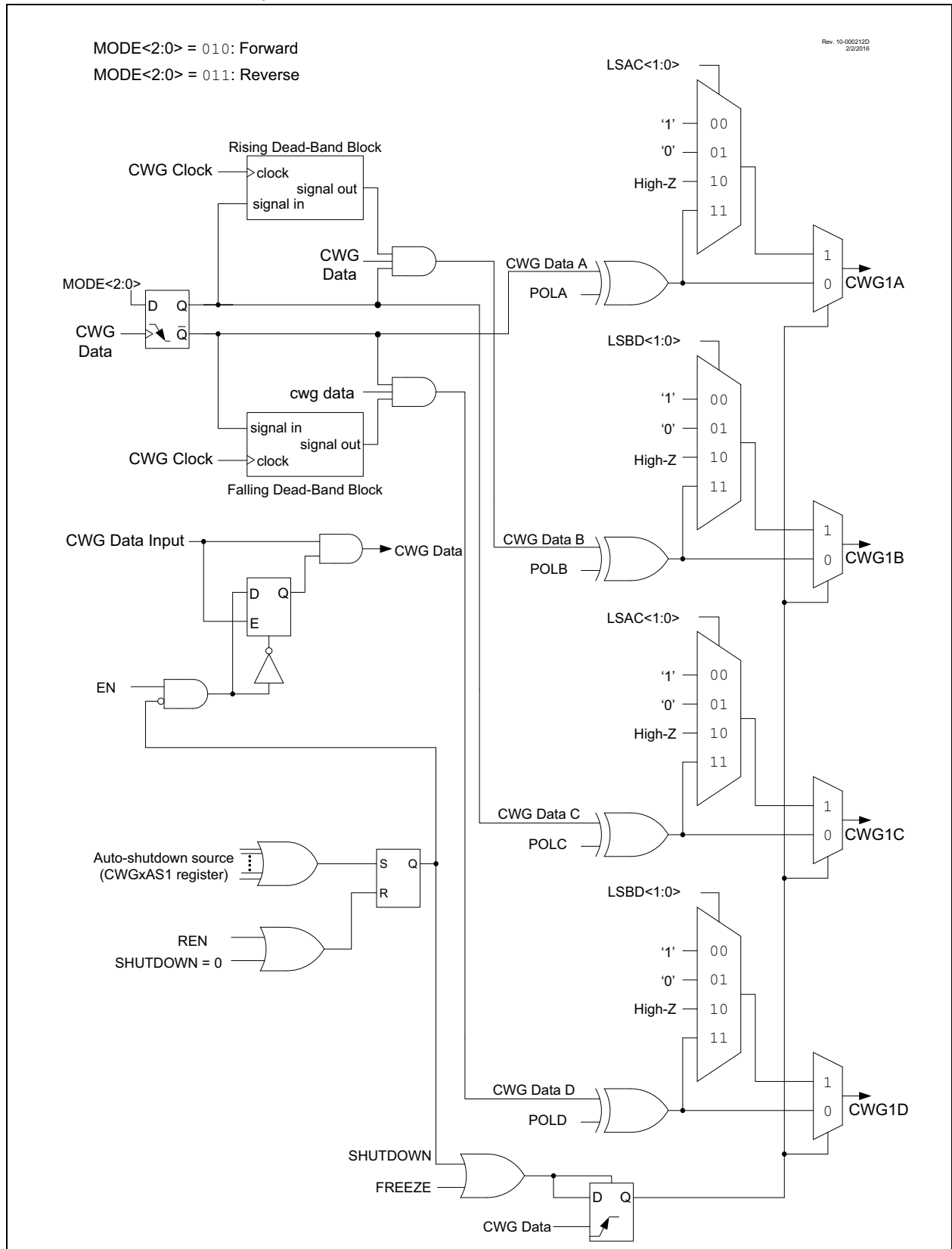


**FIGURE 26-6: SIMPLIFIED CWG BLOCK DIAGRAM (FORWARD AND REVERSE FULL-BRIDGE MODES)**

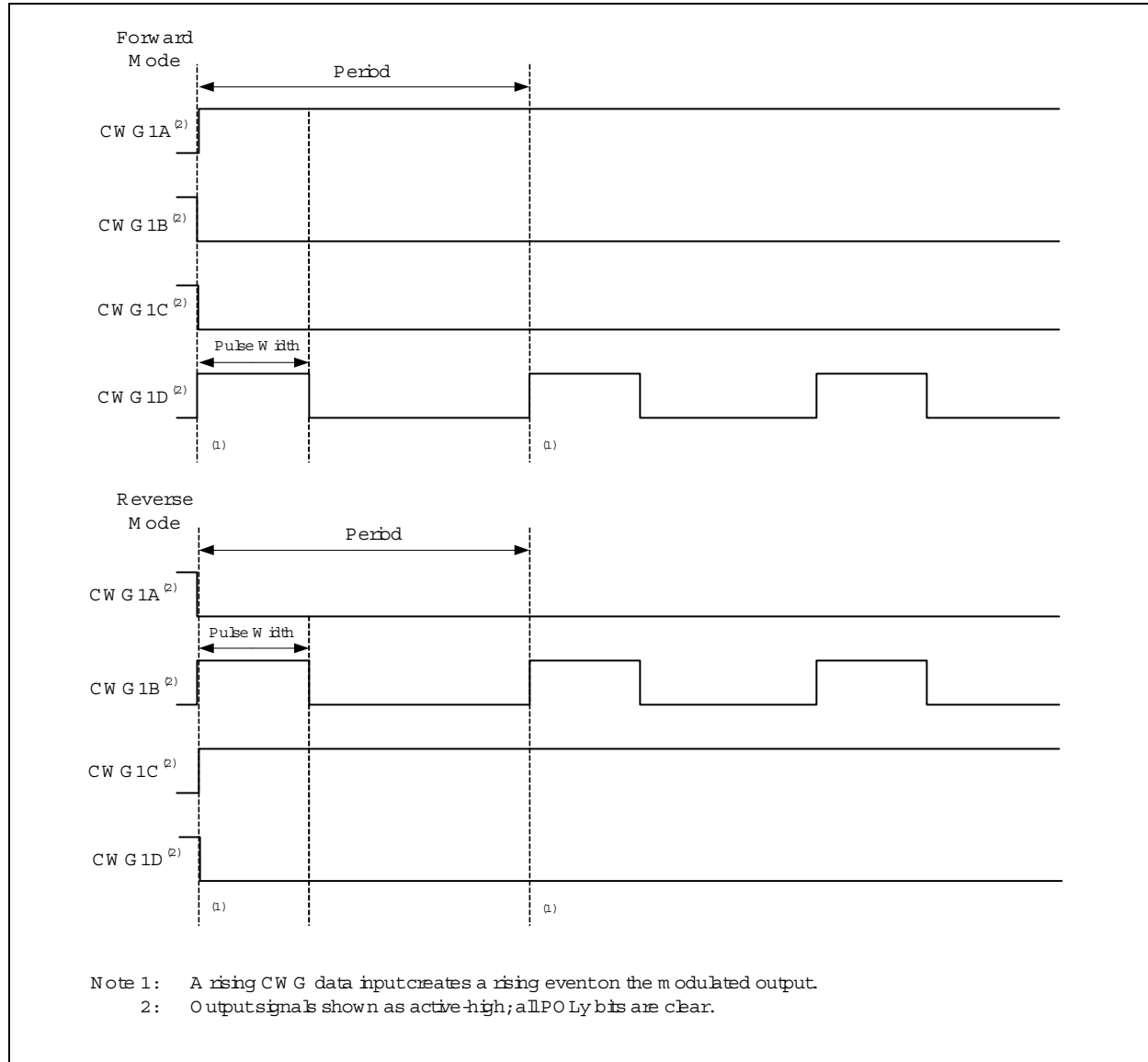


In Forward Full-Bridge mode ( $\text{MODE}\langle 2:0 \rangle = 010$ ), CWGxA is driven to its active state, CWGxB and CWGxC are driven to their inactive state, and CWGxD is modulated by the input signal, as shown in Figure 26-7.

In Reverse Full-Bridge mode ( $\text{MODE}\langle 2:0 \rangle = 011$ ), CWGxC is driven to its active state, CWGxA and CWGxD are driven to their inactive states, and CWGxB is modulated by the input signal, as shown in Figure 26-7.

In Full-Bridge mode, the dead-band period is used when there is a switch from forward to reverse or vice-versa. This dead-band control is described in [Section 26.6 “Dead-Band Control”](#), with additional details in [Section 26.7 “Rising Edge and Reverse Dead Band”](#) and [Section 26.8 “Falling Edge and Forward Dead Band”](#). Steering modes are not used with either of the Full-Bridge modes. The mode selection may be toggled between forward and reverse toggling the  $\text{MODE}\langle 0 \rangle$  bit of the CWGxCON0 while keeping  $\text{MODE}\langle 2:1 \rangle$  static, without disabling the CWG module.

**FIGURE 26-7: EXAMPLE OF FULL-BRIDGE OUTPUT**



## 26.2.3.1 Direction Change in Full-Bridge Mode

In Full-Bridge mode, changing MODE<2:0> controls the forward/reverse direction. Changes to MODE<2:0> change to the new direction on the next rising edge of the modulated input.

A direction change is initiated in software by changing the MODE<2:0> bits of the CWGxCON0 register. The sequence is illustrated in Figure 26-8.

- The associated active output CWGxA and the inactive output CWGxC are switched to drive in the opposite direction.
- The previously modulated output CWGxD is switched to the inactive state, and the previously inactive output CWGxB begins to modulate.
- CWG modulation resumes after the direction-switch dead band has elapsed.

## 26.2.3.2 Dead-Band Delay in Full-Bridge Mode

Dead-band delay is important when either of the following conditions is true:

1. The direction of the CWG output changes when the duty cycle of the data input is at or near 100%, or
2. The turn-off time of the power switch, including the power device and driver circuit, is greater than the turn-on time.

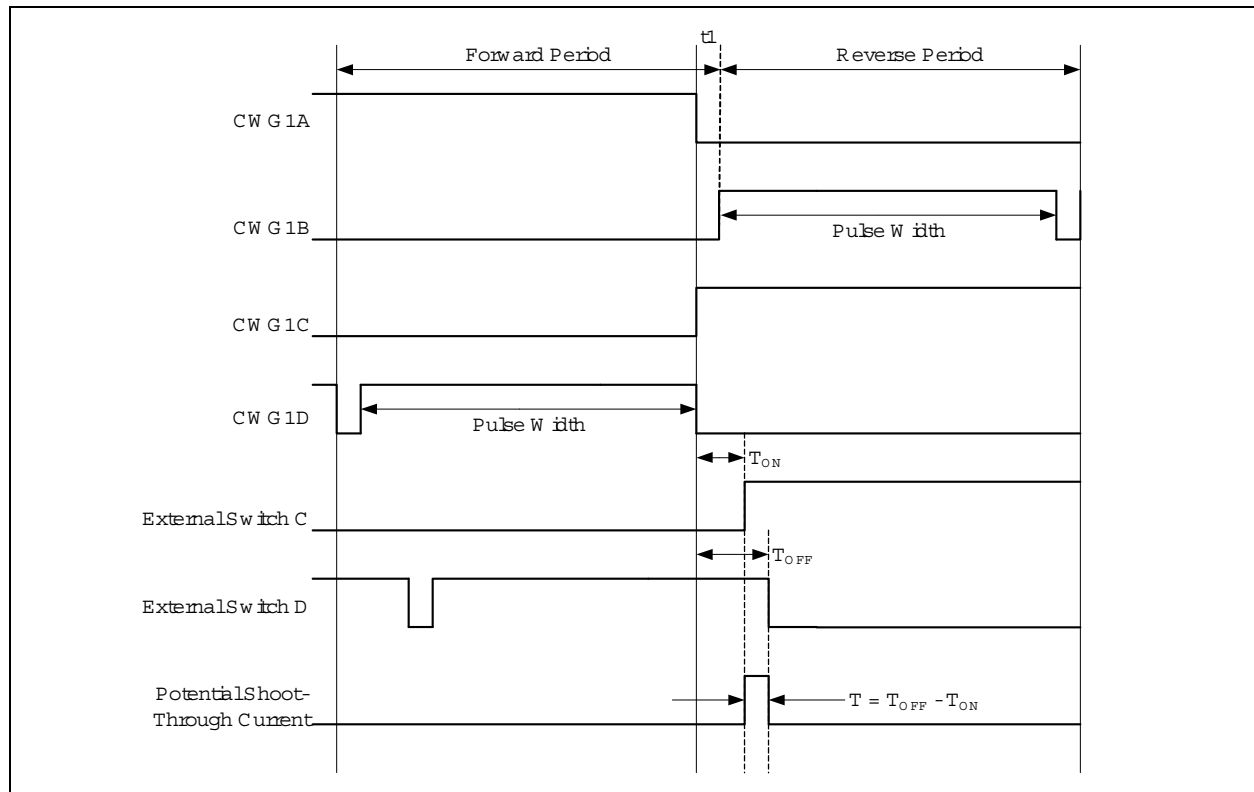
The dead-band delay is inserted only when changing directions, and only the modulated output is affected. The statically-configured outputs (CWGxA and CWGxC) are not afforded dead band, and switch essentially simultaneously.

Figure 26-8 shows an example of the CWG outputs changing directions from forward to reverse, at near 100% duty cycle. In this example, at time t1, the output of CWGxA and CWGxD become inactive, while output CWGxC becomes active. Since the turn-off time of the power devices is longer than the turn-on time, a shoot-through current will flow through power devices QC and QD for the duration of 't'. The same phenomenon will occur to power devices QA and QB for the CWG direction change from reverse to forward.

When changing the CWG direction at high duty cycle is required for an application, two possible solutions for eliminating the shoot-through current are:

1. Reduce the CWG duty cycle for one period before changing directions.
2. Use switch drivers that can drive the switches off faster than they can drive them on.

**FIGURE 26-8: EXAMPLE OF PWM DIRECTION CHANGE AT NEAR 100% DUTY CYCLE**



## 26.2.4 STEERING MODES

In both Synchronous and Asynchronous Steering modes, the modulated input signal can be steered to any combination of four CWG outputs and a fixed-value will be presented on all the outputs not used for the PWM output. Each output has independent polarity, steering, and shutdown options. Dead-band control is not used in either steering mode.

When  $STRx = 0$  (Register 26-5), then the corresponding pin is held at the level defined by  $OVRx$  (Register 26-5). When  $STRx = 1$ , then the pin is driven by the modulated input signal.

The  $POLx$  bits (Register 26-2) control the signal polarity only when  $STRx = 1$ .

The CWG auto-shutdown operation also applies to steering modes as described in Section 26.14 “Register Definitions: CWG Control”.

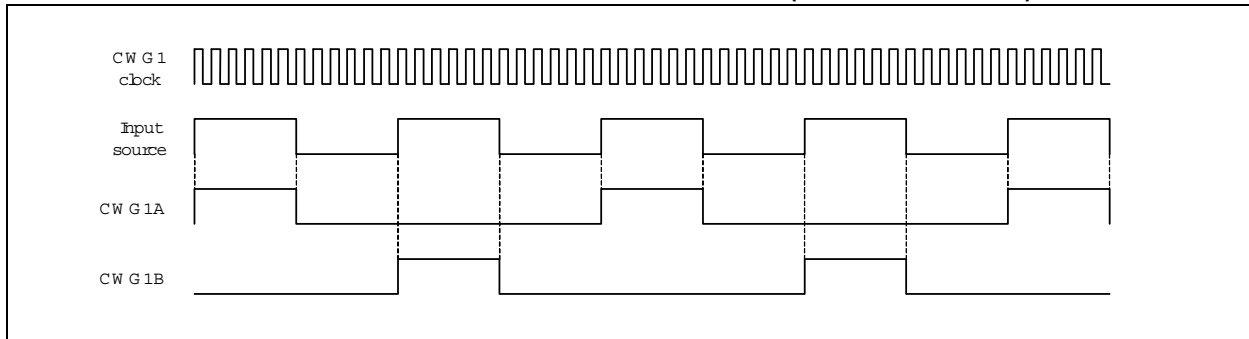
**Note:** Only the  $STRx$  bits are synchronized; the  $SDATx$  (data) bits are not synchronized.

The CWG auto-shutdown operation also applies in Steering modes as described in Section 26.10 “Auto-Shutdown”. An auto-shutdown event will only affect pins that have  $STRx = 1$ .

### 26.2.4.1 Synchronous Steering Mode

In Synchronous Steering mode ( $MODE<2:0>$  bits = 001, Register 26-1), changes to steering selection registers take effect on the next rising edge of the modulated data input (Figure 26-9). In Synchronous Steering mode, the output will always produce a complete waveform.

**FIGURE 26-9: EXAMPLE OF SYNCHRONOUS STEERING ( $MODE<2:0> = 001$ )**

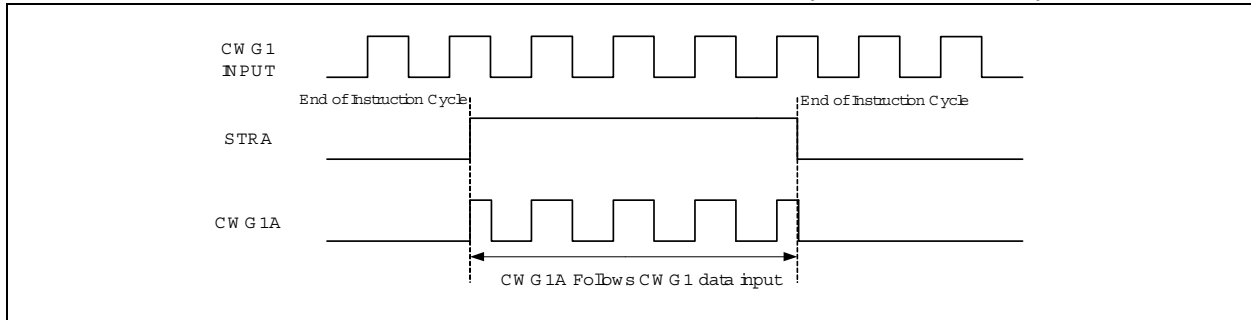




## 26.2.4.2 Asynchronous Steering Mode

In Asynchronous mode (MODE<2:0> bits = 000, [Register 26-1](#)), steering takes effect at the end of the instruction cycle that writes to STR. In Asynchronous Steering mode, the output signal may be an incomplete waveform ([Figure 26-10](#)). This operation may be useful when the user firmware needs to immediately remove a signal from the output pin.

**FIGURE 26-10: EXAMPLE OF ASYNCHRONOUS STEERING (MODE<2:0>= 000)**

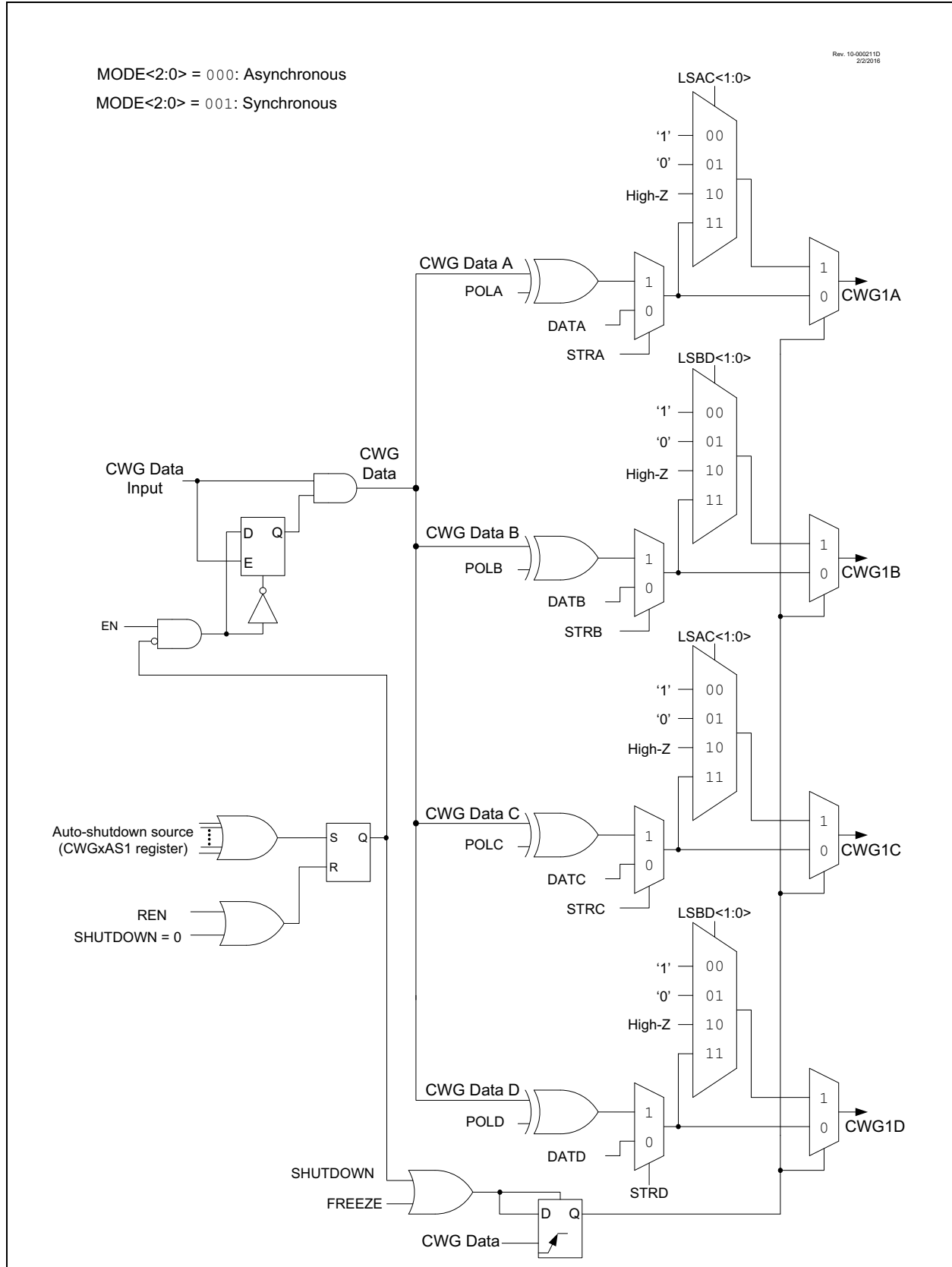


## 26.2.4.3 Start-up Considerations

The application hardware must use the proper external pull-up and/or pull-down resistors on the CWG output pins. This is required because all I/O pins are forced to high-impedance at Reset.

The POLy bits ([Register 26-2](#)) allow the user to choose whether the output signals are active-high or active-low.

**FIGURE 26-11: SIMPLIFIED CWG BLOCK DIAGRAM (OUTPUT STEERING MODES)**



## 26.3 Clock Source

The clock source is used to drive the dead-band timing circuits. The CWG module allows the following clock sources to be selected:

- FOSC (system clock)
- HFINTOSC

When the HFINTOSC is selected, the HFINTOSC will be kept running during Sleep. Therefore, CWG modes requiring dead band can operate in Sleep, provided that the CWG data input is also active during Sleep. The clock sources are selected using the CS bit of the CWGxCLKCON register ([Register 26-3](#)). The system clock FOSC, is disabled in Sleep and thus dead-band control cannot be used.

## 26.4 Selectable Input Sources

The CWG generates the output waveforms from the following input sources:

**TABLE 26-1: SELECTABLE INPUT SOURCES**

| Source Peripheral | Signal Name                  | ISM<2:0> |
|-------------------|------------------------------|----------|
| CWGxPPS           | Pin selected by CWGxPPS      | 000      |
| CCP1              | CCP1 Output                  | 001      |
| CCP2              | CCP2 Output                  | 010      |
| PWM3              | PWM3 Output                  | 011      |
| PWM4              | PWM4 Output                  | 100      |
| CMP1              | Comparator 1 Output          | 101      |
| CMP2              | Comparator 2 Output          | 110      |
| DSM               | Data signal modulator output | 111      |

The input sources are selected using the IS<4:0> bits in the CWGxISM register ([Register 26-4](#)).

## 26.5 Output Control

### 26.5.1 CWG OUTPUTS

Each CWG output can be routed to a Peripheral Pin Select (PPS) output via the RxyPPS register (see [Section 17.0 “Peripheral Pin Select \(PPS\) Module”](#)).

### 26.5.2 POLARITY CONTROL

The polarity of each CWG output can be selected independently. When the output polarity bit is set, the corresponding output is active-high. Clearing the output polarity bit configures the corresponding output as active-low. However, polarity does not affect the override levels. Output polarity is selected with the POLY bits of the CWGxCON1. Auto-shutdown and steering options are unaffected by polarity.

## 26.6 Dead-Band Control

The dead-band control provides non-overlapping PWM signals to prevent shoot-through current in PWM switches. Dead-band operation is employed for Half-Bridge and Full-Bridge modes. The CWG contains two 6-bit dead-band counters. One is used for the rising edge of the input source control in Half-Bridge mode or for reverse dead-band Full-Bridge mode. The other is used for the falling edge of the input source control in Half-Bridge mode or for forward dead band in Full-Bridge mode.

Dead band is timed by counting CWG clock periods from zero up to the value in the rising or falling dead-band counter registers. See CWGxDBR and CWGxDBF registers, respectively.

### 26.6.1 DEAD-BAND FUNCTIONALITY IN HALF-BRIDGE MODE

In Half-Bridge mode, the dead-band counters dictate the delay between the falling edge of the normal output and the rising edge of the inverted output. This can be seen in [Figure 26-2](#).

### 26.6.2 DEAD-BAND FUNCTIONALITY IN FULL-BRIDGE MODE

In Full-Bridge mode, the dead-band counters are used when undergoing a direction change. The MODE<0> bit of the CWGxCON0 register can be set or cleared while the CWG is running, allowing for changes from Forward to Reverse mode. The CWGxA and CWGxC signals will change immediately upon the first rising input edge following a direction change, but the modulated signals (CWGxB or CWGxD, depending on the direction of the change) will experience a delay dictated by the dead-band counters.

## 26.7 Rising Edge and Reverse Dead Band

In Half-Bridge mode, the rising edge dead band delays the turn-on of the CWGxA output after the rising edge of the CWG data input. In Full-Bridge mode, the reverse dead-band delay is only inserted when changing directions from Forward mode to Reverse mode, and only the modulated output CWGxB is affected.

The CWGxDBR register determines the duration of the dead-band interval on the rising edge of the input source signal. This duration is from 0 to 64 periods of the CWG clock.

Dead band is always initiated on the edge of the input source signal. A count of zero indicates that no dead band is present.

If the input source signal reverses polarity before the dead-band count is completed, then no signal will be seen on the respective output.

The CWGxDBR register value is double-buffered. When  $EN = 0$  ([Register 26-1](#)), the buffer is loaded when CWGxDBR is written. If  $EN = 1$ , then the buffer will be loaded at the rising edge following the first falling edge of the data input, after the LD bit ([Register 26-1](#)) is set. Refer to [Figure 26-12](#) for an example.

## 26.8 Falling Edge and Forward Dead Band

In Half-Bridge mode, the falling edge dead band delays the turn-on of the CWGxB output at the falling edge of the CWG data input. In Full-Bridge mode, the forward dead-band delay is only inserted when changing directions from Reverse mode to Forward mode, and only the modulated output CWGxD is affected.

The CWGxDBF register determines the duration of the dead-band interval on the falling edge of the input source signal. This duration is from zero to 64 periods of CWG clock.

Dead-band delay is always initiated on the edge of the input source signal. A count of zero indicates that no dead band is present.

If the input source signal reverses polarity before the dead-band count is completed, then no signal will be seen on the respective output.

The CWGxDBF register value is double-buffered. When  $EN = 0$  ([Register 26-1](#)), the buffer is loaded when CWGxDBF is written. If  $EN = 1$ , then the buffer will be loaded at the rising edge following the first falling edge of the data input after the LD ([Register 26-1](#)) is set. Refer to [Figure 26-13](#) for an example.

**FIGURE 26-12: DEAD-BAND OPERATION, CWGxDBR = 0x01, CWGxDBF = 0x02**

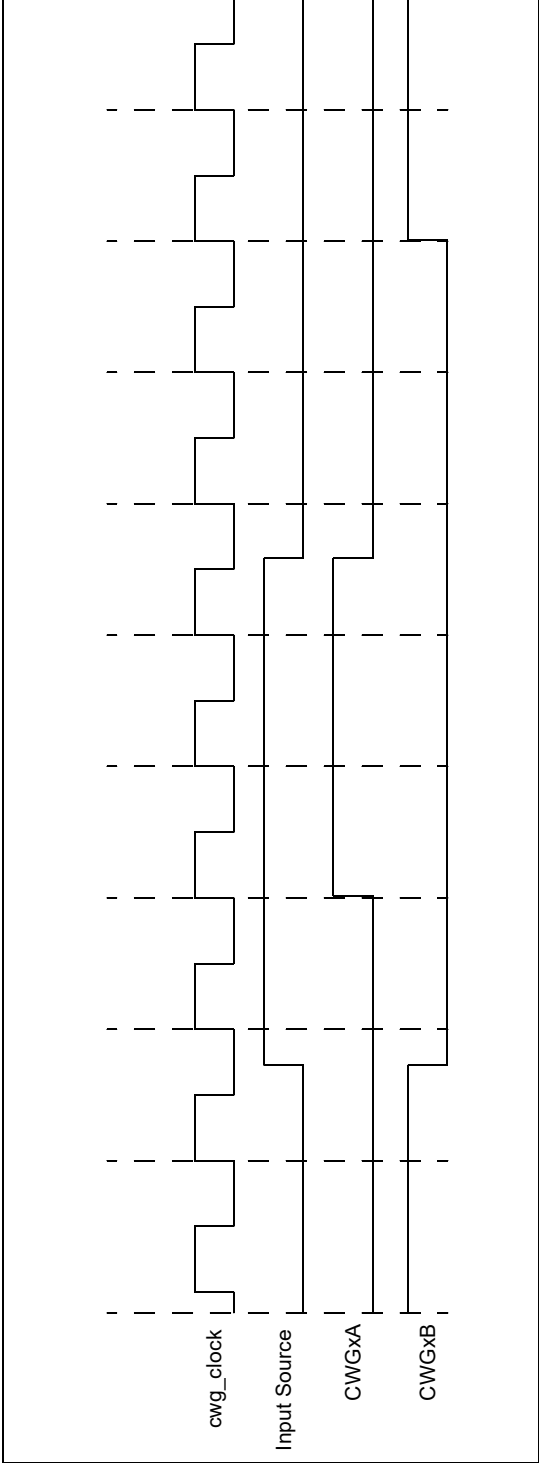
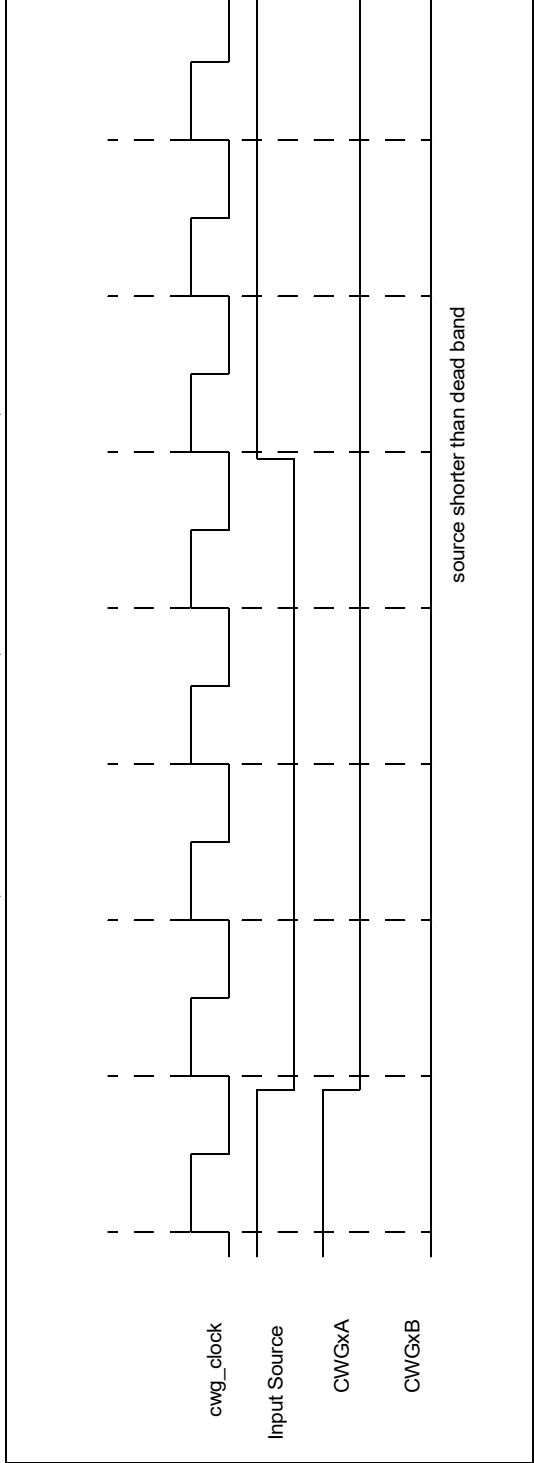


**FIGURE 26-13: DEAD-BAND OPERATION, CWGxDBR = 0x03, CWGxDBF = 0x06, SOURCE SHORTER THAN DEAD BAND**



## 26.9 Dead-Band Jitter

When the rising and falling edges of the input source are asynchronous to the CWG clock, it creates jitter in the dead-band time delay. The maximum jitter is equal to one CWG clock period. Refer to [Equation 26-1](#) for more details.

### EQUATION 26-1: DEAD-BAND DELAY TIME CALCULATION

$$T_{\text{DEAD-BAND\_MIN}} = \frac{1}{F_{\text{CWG\_CLOCK}}} \bullet \text{DBx} < 4:0 >$$

$$T_{\text{DEAD-BAND\_MAX}} = \frac{1}{F_{\text{CWG\_CLOCK}}} \bullet \text{DBx} < 4:0 > + 1$$

$$T_{\text{JITTER}} = T_{\text{DEAD-BAND\_MAX}} - T_{\text{DEAD-BAND\_MIN}}$$

$$T_{\text{JITTER}} = \frac{1}{F_{\text{CWG\_CLOCK}}}$$

$$T_{\text{DEAD-BAND\_MAX}} = T_{\text{DEAD-BAND\_MIN}} + T_{\text{JITTER}}$$

EXAMPLE

$$\text{DBR} < 4:0 > = 0x0A = 10$$

$$F_{\text{CWG\_CLOCK}} = 8 \text{ MHz}$$

$$T_{\text{JITTER}} = \frac{1}{8 \text{ MHz}} = 125 \text{ ns}$$

$$T_{\text{DEAD-BAND\_MIN}} = 125 \text{ ns} \times 10 = 125 \text{ } \mu\text{s}$$

$$T_{\text{DEAD-BAND\_MAX}} = 125 \text{ } \mu\text{s} + 0.125 \text{ } \mu\text{s} = 1.37 \text{ } \mu\text{s}$$

## 26.10 Auto-Shutdown

Auto-shutdown is a method to immediately override the CWG output levels with specific overrides that allow for safe shutdown of the circuit. The shutdown state can be either cleared automatically or held until cleared by software. The auto-shutdown circuit is illustrated in [Figure 26-14](#).

### 26.10.1 SHUTDOWN

The shutdown state can be entered by either of the following two methods:

- Software generated
- External Input

#### 26.10.1.1 Software Generated Shutdown

Setting the SHUTDOWN bit of the CWGxAS0 register will force the CWG into the shutdown state.

When the auto-restart is disabled, the shutdown state will persist as long as the SHUTDOWN bit is set.

When auto-restart is enabled, the SHUTDOWN bit will clear automatically and resume operation on the next rising edge event. The SHUTDOWN bit indicates when a shutdown condition exists. The bit may be set or cleared in software or by hardware.

#### 26.10.1.2 External Input Source

External shutdown inputs provide the fastest way to safely suspend CWG operation in the event of a Fault condition. When any of the selected shutdown inputs goes active, the CWG outputs will immediately go to the specified override levels without software delay. The override levels are selected by the LSB<1:0> and LSAC<1:0> bits of the CWGxAS0 register ([Register 26-6](#)). Several input sources can be selected to cause a shutdown condition. All input sources are active-low. The sources are:

- Pin selected by CWGxPPS
- Timer2 postscaled output
- Timer4 postscaled output
- Timer6 postscaled output
- Comparator 1 output
- Comparator 2 output
- CLC2 output

Shutdown input sources are individually enabled by the ASxE bits of the CWGxAS1 register ([Register 26-7](#)).

**Note:** Shutdown inputs are level sensitive, not edge sensitive. The shutdown state cannot be cleared, except by disabling auto-shutdown, as long as the shutdown input level persists.

#### 26.10.1.3 Pin Override Levels

The levels driven to the CWG outputs during an auto-shutdown event are controlled by the LSB<1:0> and LSAC<1:0> bits of the CWGxAS0 register ([Register 26-6](#)). The LSB<1:0> bits control CWGxB/D output levels, while the LSAC<1:0> bits control the CWGxA/C output levels.

#### 26.10.1.4 Auto-Shutdown Interrupts

When an auto-shutdown event occurs, either by software or hardware setting SHUTDOWN, the CWGxIF flag bit of the respective PIR register is set.

## 26.11 Auto-Shutdown Restart

After an auto-shutdown event has occurred, there are two ways to resume operation:

- Software controlled
- Auto-restart

In either case, the shutdown source must be cleared before the restart can take place. That is, either the shutdown condition must be removed, or the corresponding ASxE bit must be cleared.

### 26.11.1 SOFTWARE-CONTROLLED RESTART

If the REN bit of the CWGxAS0 register is clear (REN = 0), the CWG module must be restarted after an auto-shutdown event through software.

Once all auto-shutdown sources are removed, the software must clear SHUTDOWN. Once SHUTDOWN is cleared, the CWG module will resume operation upon the first rising edge of the CWG data input.

**Note:** The SHUTDOWN bit cannot be cleared in software if the auto-shutdown condition is still present.

### 26.11.2 AUTO-RESTART

If the REN bit of the CWGxAS0 register is set (REN = 1), the CWG module will restart from the shutdown state automatically.

Once all auto-shutdown conditions are removed, the hardware will automatically clear SHUTDOWN. Once SHUTDOWN is cleared, the CWG module will resume operation upon the first rising edge of the CWG data input.

**Note:** The SHUTDOWN bit cannot be cleared in software if the auto-shutdown condition is still present.

## 26.12 Operation During Sleep

The CWG module operates independently from the system clock and will continue to run during Sleep, provided that the clock and input sources selected remain active.

The HFINTOSC remains active during Sleep when all the following conditions are met:

- CWG module is enabled
- Input source is active
- HFINTOSC is selected as the clock source, regardless of the system clock source selected.

In other words, if the HFINTOSC is simultaneously selected as system clock and CWG clock, when the CWG is enabled and the input source is active, then the CPU will go Idle during Sleep, but the HFINTOSC will remain active and the CWG will continue to operate. This will have a direct effect on the Sleep mode current.

## 26.13 Configuring the CWG

1. Ensure that the TRIS control bits corresponding to CWG outputs are set so that all are configured as inputs, ensuring that the outputs are inactive during setup. External hardware should ensure that pin levels are held to safe levels.
2. Clear the EN bit, if not already cleared.
3. Configure the MODE<2:0> bits of the CWGxCON0 register to set the output operating mode.
4. Configure the POLY bits of the CWGxCON1 register to set the output polarities.
5. Configure the ISM<4:0> bits of the CWGxISM register to select the data input source.
6. If a steering mode is selected, configure the STRx bits to select the desired output on the CWG outputs.
7. Configure the LSB<1:0> and LSAC<1:0> bits of the CWGxASD0 register to select the auto-shutdown output override states (this is necessary even if not using auto-shutdown because start-up will be from a shutdown state).
8. If auto-restart is desired, set the REN bit of CWGxAS0.
9. If auto-shutdown is desired, configure the ASxE bits of the CWGxAS1 register to select the shutdown source.
10. Set the desired rising and falling dead-band times with the CWGxDBR and CWGxDBF registers.
11. Select the clock source in the CWGxCLKCON register.
12. Set the EN bit to enable the module.
13. Clear the TRIS bits that correspond to the CWG outputs to set them as outputs.

If auto-restart is to be used, set the REN bit and the SHUTDOWN bit will be cleared automatically. Otherwise, clear the SHUTDOWN bit in software to start the CWG.



**FIGURE 26-14: CWG SHUTDOWN BLOCK DIAGRAM**

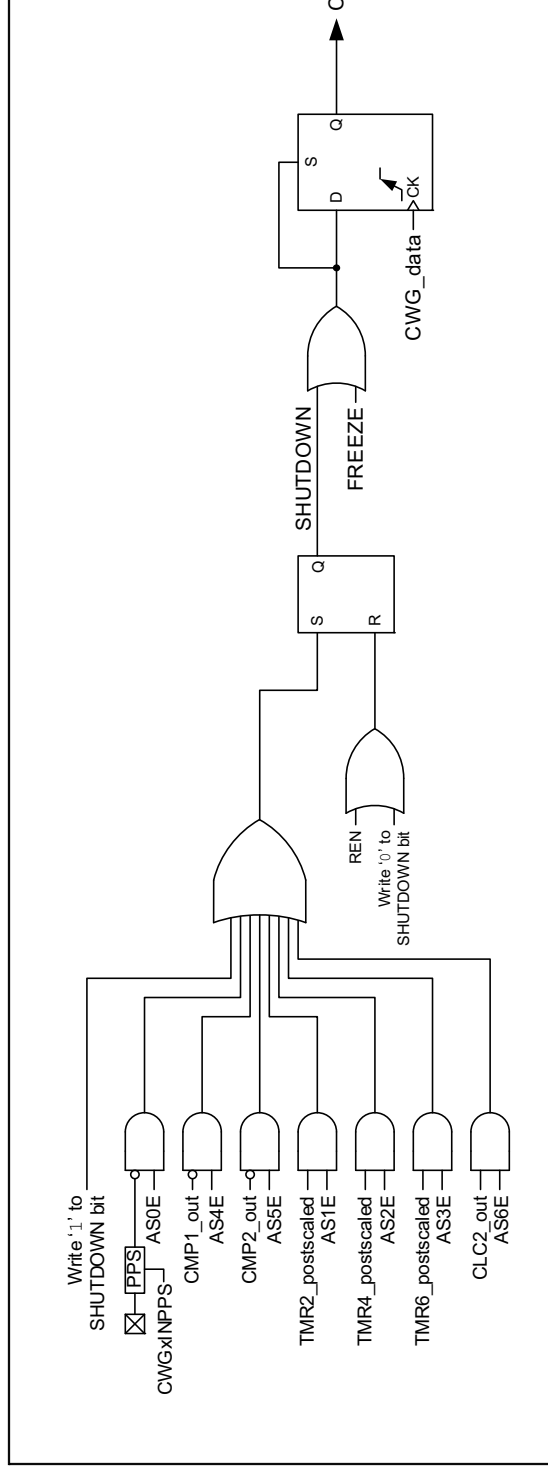


**FIGURE 26-15: SHUTDOWN FUNCTIONALITY, AUTO-RESTART DISABLED (REN = 0, LSAC = 01, LSBSD = 01)**

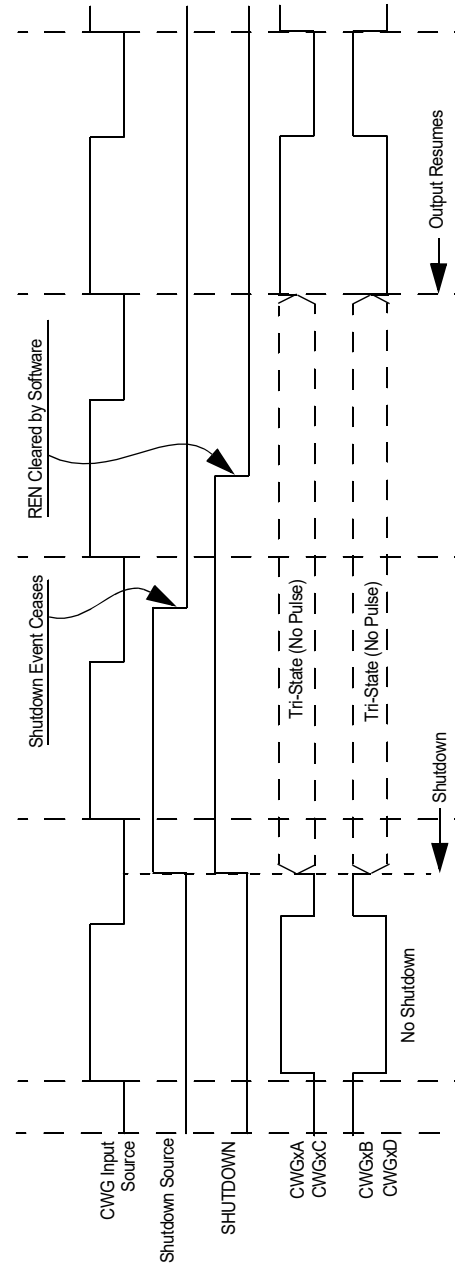
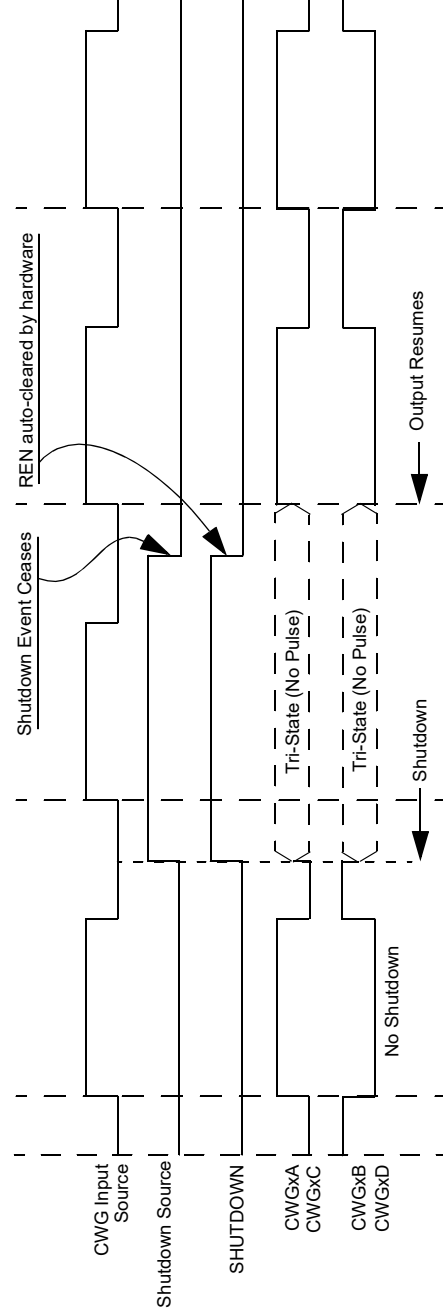


**FIGURE 26-16: SHUTDOWN FUNCTIONALITY, AUTO-RESTART ENABLED (REN = 1, LSAC = 01, LSB D = 01)**



## 26.14 Register Definitions: CWG Control

Long bit name prefixes for the CWG peripheral is shown below. Refer to [Section 1.3.2.2 “Long Bit Names”](#) for more information.

| Peripheral | Bit Name Prefix |
|------------|-----------------|
| CWG1       | CWG1            |
| CWG2       | CWG2            |
| CWG3       | CWG3            |

### REGISTER 26-1: CWGxCON0: CWG CONTROL REGISTER 0

| R/W-0/0 | R/W/HC-0/0        | U-0 | U-0 | U-0 | R/W-0/0   | R/W-0/0 | R/W-0/0 |
|---------|-------------------|-----|-----|-----|-----------|---------|---------|
| EN      | LD <sup>(1)</sup> | —   | —   | —   | MODE<2:0> |         |         |
| bit 7   |                   |     |     |     |           |         | bit 0   |

#### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

HC = Bit is cleared by hardware

bit 7

**EN:** CWGx Enable bit

1 = Module is enabled

0 = Module is disabled

bit 6

**LD:** CWGx Load Buffers bit<sup>(1)</sup>

1 = Dead-band count buffers to be loaded on CWG data rising edge, following first falling edge after this bit is set

0 = Buffers remain unchanged

bit 5-3

**Unimplemented:** Read as '0'

bit 2-0

**MODE<2:0>:** CWGx Mode bits

111 = Reserved

110 = Reserved

101 = CWG outputs operate in Push-Pull mode

100 = CWG outputs operate in Half-Bridge mode

011 = CWG outputs operate in Reverse Full-Bridge mode

010 = CWG outputs operate in Forward Full-Bridge mode

001 = CWG outputs operate in Synchronous Steering mode

000 = CWG outputs operate in Asynchronous Steering mode

**Note 1:** This bit can only be set after EN = 1; it cannot be set in the same cycle when EN is set.

**REGISTER 26-2: CWGxCON1: CWG CONTROL REGISTER 1**

|       |     |     |     |         |         |         |         |
|-------|-----|-----|-----|---------|---------|---------|---------|
| U-0   | U-0 | R-x | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| —     | —   | IN  | —   | POLD    | POLC    | POLB    | POLA    |
| bit 7 |     |     |     | bit 0   |         |         |         |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

q = Value depends on condition

- bit 7-6      **Unimplemented:** Read as '0'
- bit 5        **IN:** CWG Input Value bit (read-only)
- bit 4        **Unimplemented:** Read as '0'
- bit 3        **POLD:** CWGxD Output Polarity bit  
1 = Signal output is inverted polarity  
0 = Signal output is normal polarity
- bit 2        **POLC:** CWGxC Output Polarity bit  
1 = Signal output is inverted polarity  
0 = Signal output is normal polarity
- bit 1        **POLB:** CWGxB Output Polarity bit  
1 = Signal output is inverted polarity  
0 = Signal output is normal polarity
- bit 0        **POLA:** CWGxA Output Polarity bit  
1 = Signal output is inverted polarity  
0 = Signal output is normal polarity

**REGISTER 26-3: CWGxCLK: CWGx CLOCK INPUT SELECTION REGISTER**

|       |     |     |     |     |     |     |         |
|-------|-----|-----|-----|-----|-----|-----|---------|
| U-0   | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0/0 |
| —     | —   | —   | —   | —   | —   | —   | CS      |
| bit 7 |     |     |     |     |     |     | bit 0   |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

q = Value depends on condition

bit 7-1

**Unimplemented:** Read as '0'

bit 0

**CS:** CWG Clock Source Selection Select bits

| CS | CWG1                    | CWG2                    | CWG3                    |
|----|-------------------------|-------------------------|-------------------------|
| 1  | HFINTOSC <sup>(1)</sup> | HFINTOSC <sup>(1)</sup> | HFINTOSC <sup>(1)</sup> |
| 0  | Fosc                    | Fosc                    | Fosc                    |

**Note 1:** HFINTOSC remains operating during Sleep.

**REGISTER 26-4: CWGxISM: CWGx INPUT SELECTION REGISTER**

|       |     |     |         |         |         |         |         |
|-------|-----|-----|---------|---------|---------|---------|---------|
| U-0   | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| —     | —   | —   | IS<4:0> |         |         |         |         |
| bit 7 |     |     | bit 0   |         |         |         |         |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

q = Value depends on condition

bit 7-5

**Unimplemented** Read as '0'

bit 4-0

**IS<4:0>:** CWG Data Input Selection Multiplexer Select bits

| IS<4:0>     | CWG1                    | CWG2                    | CWG3                    |
|-------------|-------------------------|-------------------------|-------------------------|
|             | Input Selection         | Input Selection         | Input Selection         |
| 11111-10011 | Reserved                | Reserved                | Reserved                |
| 10010       | CLC4_out                | CLC4_out                | CLC4_out                |
| 10001       | CLC3_out                | CLC3_out                | CLC3_out                |
| 10000       | CLC2_out                | CLC2_out                | CLC2_out                |
| 01111       | CLC1_out                | CLC1_out                | CLC1_out                |
| 01110       | DSM_out                 | DSM_out                 | DSM_out                 |
| 01101       | CMP2OUT                 | CMP2OUT                 | CMP2OUT                 |
| 01100       | CMP1OUT                 | CMP1OUT                 | CMP1OUT                 |
| 01011       | NCO1OUT                 | NCO1OUT                 | NCO1OUT                 |
| 01010-01001 | Reserved                | Reserved                | Reserved                |
| 01000       | PWM8OUT                 | PWM8OUT                 | PWM8OUT                 |
| 00111       | PWM7OUT                 | PWM7OUT                 | PWM7OUT                 |
| 00110       | PWM6OUT                 | PWM6OUT                 | PWM6OUT                 |
| 00101       | PWM5OUT                 | PWM5OUT                 | PWM5OUT                 |
| 00100       | CCP4_out                | CCP4_out                | CCP4_out                |
| 00011       | CCP3_out                | CCP3_out                | CCP3_out                |
| 00010       | CCP2_out                | CCP2_out                | CCP2_out                |
| 00001       | CCP1_out                | CCP1_out                | CCP1_out                |
| 00000       | Pin selected by CWG1PPS | Pin selected by CWG2PPS | Pin selected by CWG3PPS |

## REGISTER 26-5: CWGxSTR<sup>(1)</sup>: CWG STEERING CONTROL REGISTER

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0             | R/W-0/0             | R/W-0/0             | R/W-0/0             |
|---------|---------|---------|---------|---------------------|---------------------|---------------------|---------------------|
| OVRD    | OVRC    | OVRB    | OVRA    | STRD <sup>(2)</sup> | STRC <sup>(2)</sup> | STRB <sup>(2)</sup> | STRA <sup>(2)</sup> |
| bit 7   |         |         |         |                     |                     |                     | bit 0               |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

q = Value depends on condition

bit 7 **OVRD**: Steering Data D bit

bit 6 **OVRC**: Steering Data C bit

bit 5 **OVRB**: Steering Data B bit

bit 4 **OVRA**: Steering Data A bit

bit 3 **STRD**: Steering Enable bit D<sup>(2)</sup>

1 = CWGxD output has the CWG data input waveform with polarity control from POLD bit

0 = CWGxD output is assigned to value of OVRD bit

bit 2 **STRC**: Steering Enable bit C<sup>(2)</sup>

1 = CWGxC output has the CWG data input waveform with polarity control from POLC bit

0 = CWGxC output is assigned to value of OVRC bit

bit 1 **STRB**: Steering Enable bit B<sup>(2)</sup>

1 = CWGxB output has the CWG data input waveform with polarity control from POLB bit

0 = CWGxB output is assigned to value of OVRB bit

bit 0 **STRA**: Steering Enable bit A<sup>(2)</sup>

1 = CWGxA output has the CWG data input waveform with polarity control from POLA bit

0 = CWGxA output is assigned to value of OVRA bit

**Note 1:** The bits in this register apply only when MODE<2:0> = 00x ([Register 26-1](#), Steering modes).

**2:** This bit is double-buffered when MODE<2:0> = 001.

## REGISTER 26-6: CWGxAS0: CWG AUTO-SHUTDOWN CONTROL REGISTER 0

| R/W/HS/HC-0/0 | R/W-0/0 | R/W-0/0   | R/W-1/1 | R/W-0/0   | R/W-1/1 | U-0 | U-0   |
|---------------|---------|-----------|---------|-----------|---------|-----|-------|
| SHUTDOWN      | REN     | LSBD<1:0> |         | LSAC<1:0> |         | —   | —     |
| bit 7         |         |           |         |           |         |     | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

HS/HC = Bit is set/cleared by hardware

q = Value depends on condition

bit 7 **SHUTDOWN:** Auto-Shutdown Event Status bit<sup>(1,2)</sup>

1 = An auto-shutdown state is in effect

0 = No auto-shutdown event has occurred

bit 6 **REN:** Auto-Restart Enable bit

1 = Auto-restart is enabled

0 = Auto-restart is disabled

bit 5-4 **LSBD<1:0>:** CWGxB and CWGxD Auto-Shutdown State Control bits

11 = A logic '1' is placed on CWGxB/D when an auto-shutdown event occurs.

10 = A logic '0' is placed on CWGxB/D when an auto-shutdown event occurs.

01 = Pin is tri-stated on CWGxB/D when an auto-shutdown event occurs.

00 = The inactive state of the pin, including polarity, is placed on CWGxB/D after the required dead-band interval when an auto-shutdown event occurs.

bit 3-2 **LSAC<1:0>:** CWGxA and CWGxC Auto-Shutdown State Control bits

11 = A logic '1' is placed on CWGxA/C when an auto-shutdown event occurs.

10 = A logic '0' is placed on CWGxA/C when an auto-shutdown event occurs.

01 = Pin is tri-stated on CWGxA/C when an auto-shutdown event occurs.

00 = The inactive state of the pin, including polarity, is placed on CWGxA/C after the required dead-band interval when an auto-shutdown event occurs.

bit 1-0 **Unimplemented:** Read as '0'

**Note 1:** This bit may be written while EN = 0 ([Register 26-1](#)), to place the outputs into the shutdown configuration.

**2:** The outputs will remain in auto-shutdown state until the next rising edge of the CWG data input after this bit is cleared.



## REGISTER 26-7: CWGxAS1: CWG AUTO-SHUTDOWN CONTROL REGISTER 1

| U-0   | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|-------|---------|---------|---------|---------|---------|---------|---------|
| —     | AS6E    | AS5E    | AS4E    | AS3E    | AS2E    | AS1E    | AS0E    |
| bit 7 |         |         |         |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

q = Value depends on condition

bit 7 **Unimplemented** Read as '0'

bit 6 **AS6E:** CWG Auto-shutdown Source 6 Enable bit

1 = Auto-shutdown for Source 6 is enabled

| CWG Module             | CWG1     | CWG2     | CWG3     |
|------------------------|----------|----------|----------|
| Auto-shutdown Source 6 | CLC2 OUT | CLC3 OUT | CLC4 OUT |

0 = Auto-shutdown for Source 6 is disabled

bit 5 **AS5E:** CWG Auto-shutdown Source 5 (CMP2 OUT) Enable bit

1 = Auto-shutdown for CMP2 OUT is enabled

0 = Auto-shutdown for CMP2 OUT is disabled

bit 4 **AS4E:** CWG Auto-shutdown Source 4 (CMP1 OUT) Enable bit

1 = Auto-shutdown for CMP1 OUT is enabled

0 = Auto-shutdown for CMP1 OUT is disabled

bit 3 **AS3E:** CWG Auto-shutdown Source 3 (TMR6\_Postscaled) Enable bit

1 = Auto-shutdown for TMR6\_Postscaled is enabled

0 = Auto-shutdown for TMR6\_Postscaled is disabled

bit 2 **AS2E:** CWG Auto-shutdown Source 2 (TMR4\_Postscaled) Enable bit

1 = Auto-shutdown for TMR4\_Postscaled is enabled

0 = Auto-shutdown for TMR4\_Postscaled is disabled

bit 1 **AS1E:** CWG Auto-shutdown Source 1 (TMR2\_Postscaled) Enable bit

1 = Auto-shutdown for TMR2\_Postscaled is enabled

0 = Auto-shutdown for TMR2\_Postscaled is disabled

bit 0 **AS0E:** CWG Auto-shutdown Source 0 (Pin selected by CWGxPPS) Enable bit

1 = Auto-shutdown for CWGxPPS Pin is enabled

0 = Auto-shutdown for CWGxPPS Pin is disabled

## REGISTER 26-8: CWGxDBR: CWG RISING DEAD-BAND COUNT REGISTER

| U-0   | U-0 | R/W-x/u  | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
|-------|-----|----------|---------|---------|---------|---------|---------|
| —     | —   | DBR<5:0> |         |         |         |         |         |
| bit 7 |     |          |         |         |         |         | bit 0   |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared | q = Value depends on condition                        |

bit 7-6 **Unimplemented:** Read as '0'

bit 5-0 **DBR<5:0>:** CWG Rising Edge Triggered Dead-Band Count bits

11 1111 = 63-64 CWG clock periods

11 1110 = 62-63 CWG clock periods

.

.

.

00 0010 = 2-3 CWG clock periods

00 0001 = 1-2 CWG clock periods

00 0000 = 0 CWG clock periods. Dead-band generation is by-passed

## REGISTER 26-9: CWGxDBF: CWG FALLING DEAD-BAND COUNT REGISTER

| U-0   | U-0 | R/W-x/u  | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
|-------|-----|----------|---------|---------|---------|---------|---------|
| —     | —   | DBF<5:0> |         |         |         |         |         |
| bit 7 |     |          |         |         |         |         | bit 0   |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared | q = Value depends on condition                        |

bit 7-6 **Unimplemented:** Read as '0'

bit 5-0 **DBF<5:0>:** CWG Falling Edge Triggered Dead-Band Count bits

11 1111 = 63-64 CWG clock periods

11 1110 = 62-63 CWG clock periods

.

.

.

00 0010 = 2-3 CWG clock periods

00 0001 = 1-2 CWG clock periods

00 0000 = 0 CWG clock periods. Dead-band generation is by-passed.

**TABLE 26-2: SUMMARY OF REGISTERS ASSOCIATED WITH CWG**

| Name     | Bit 7    | Bit 6 | Bit 5     | Bit 4 | Bit 3     | Bit 2     | Bit 1 | Bit 0 | Register on Page |
|----------|----------|-------|-----------|-------|-----------|-----------|-------|-------|------------------|
| CWGxCON0 | EN       | LD    | —         | —     | —         | MODE<2:0> |       |       | 411              |
| CWGxCON1 | —        | —     | IN        | —     | POLD      | POLC      | POLB  | POLA  | 412              |
| CWGxCLK  | —        | —     | —         | —     | —         | —         | —     | CS    | 413              |
| CWGxISM  | —        | —     | —         | —     | —         | ISM<2:0>  |       |       | 414              |
| CWGxSTR  | OVRD     | OVRC  | OVRB      | OVRA  | STRD      | STRC      | STRB  | STRA  | 415              |
| CWGxAS0  | SHUTDOWN | REN   | LSBD<1:0> |       | LSAC<1:0> |           | —     | —     | 416              |
| CWGxAS1  | —        | AS6E  | AS5E      | AS4E  | AS3E      | AS2E      | AS1E  | AS0E  | 417              |
| CWGxDBR  | —        | —     | DBR<5:0>  |       |           |           |       |       | 418              |
| CWGxDBF  | —        | —     | DBF<5:0>  |       |           |           |       |       | 418              |

**Legend:** — = unimplemented locations read as '0'. Shaded cells are not used by CWG.

## 27.0 CONFIGURABLE LOGIC CELL (CLC)

The Configurable Logic Cell (CLCx) module provides programmable logic that operates outside the speed limitations of software execution. The logic cell takes up to 32 input signals and, through the use of configurable gates, reduces the 32 inputs to four logic lines that drive one of eight selectable single-output logic functions.

Input sources are a combination of the following:

- I/O pins
- Internal clocks
- Peripherals
- Register bits

The output can be directed internally to peripherals and to an output pin.

There are four CLC modules available on this device - CLC1, CLC2, CLC3 and CLC4.

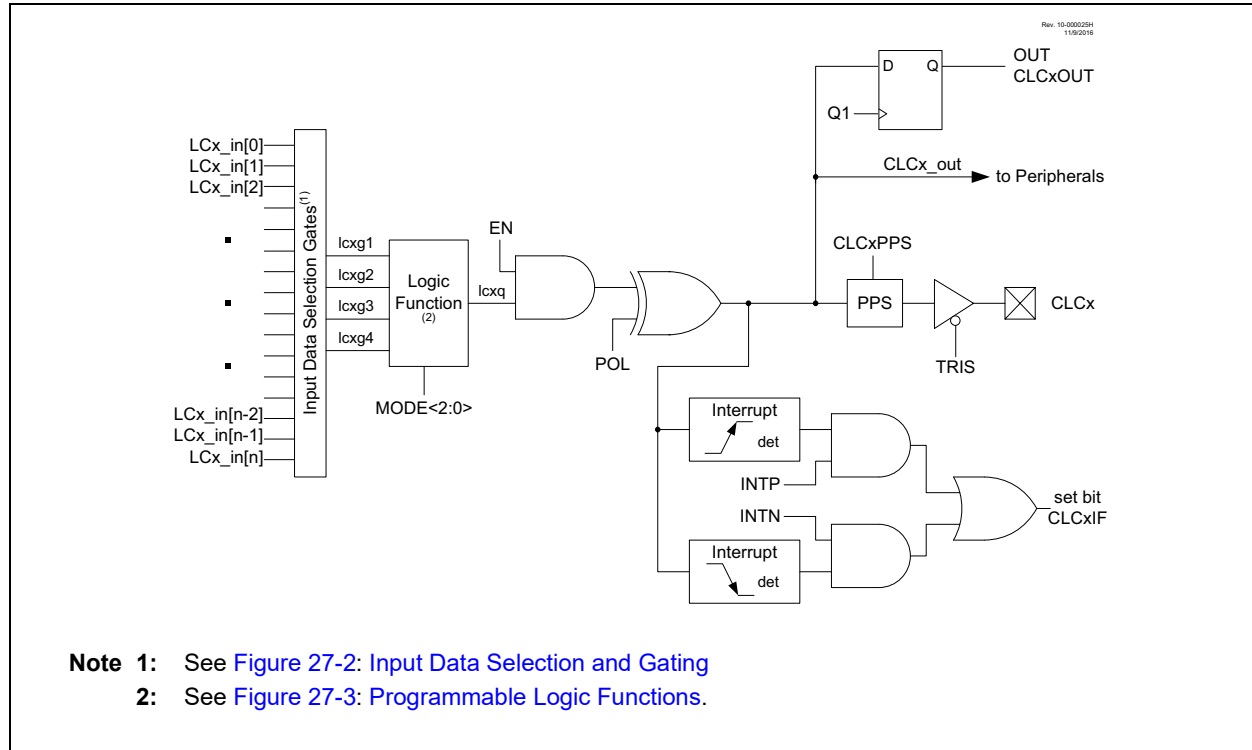
**Note:** The CLC1, CLC2, CLC3 and CLC4 are four separate module instances of the same CLC module design. Throughout this section, the lower case 'x' in register names is a generic reference to the CLC number (which should be substituted with 1, 2, 3, or 4 during code development). For example, the control register is generically described in this chapter as CLCxCON, but the actual device registers are CLC1CON, CLC2CON, CLC3CON and CLC4CON.

Refer to [Figure 27-1](#) for a simplified diagram showing signal flow through the CLCx.

Possible configurations include:

- Combinatorial Logic
  - AND
  - NAND
  - AND-OR
  - AND-OR-INVERT
  - OR-XOR
  - OR-XNOR
- Latches
  - S-R
  - Clocked D with Set and Reset
  - Transparent D with Set and Reset

**FIGURE 27-1: CLCx SIMPLIFIED BLOCK DIAGRAM**



## 27.1 CLCx Setup

Programming the CLCx module is performed by configuring the four stages in the logic signal flow. The four stages are:

- Data selection
- Data gating
- Logic function selection
- Output polarity

Each stage is setup at run time by writing to the corresponding CLCx Special Function Registers. This has the added advantage of permitting logic reconfiguration on-the-fly during program execution.

### 27.1.1 DATA SELECTION

There are 32 signals available as inputs to the configurable logic. Four 32-input multiplexers are used to select the inputs to pass on to the next stage.

Data selection is through four multiplexers as indicated on the left side of [Figure 27-2](#). Data inputs in the figure are identified by a generic numbered input name.

[Table 27-1](#) correlates the generic input name to the actual signal for each CLCx module. The column labeled 'DyS<4:0> Value' indicates the MUX selection code for the selected data input. DyS is an abbreviation for the MUX select input codes: D1S<4:0> through D4S<4:0>.

Data inputs are selected with CLCxSEL0 through CLCxSEL3 registers ([Register 27-3](#) through [Register 27-6](#)).

**Note:** Data selections are undefined at power-up.

**TABLE 27-1: CLCx DATA INPUT SELECTION**

| DyS<5:0><br>Value | CLCx Input Source |
|-------------------|-------------------|
| 111111 [63]       | Reserved          |
| •                 |                   |
| •                 |                   |
| •                 |                   |
| 110110 [55]       |                   |
| 110110 [54]       | CAN_tx1           |
| 110101 [53]       | CAN_tx0           |
| 110100 [52]       | CWG3B_out         |
| 110011 [51]       | CWG3A_out         |
| 110010 [50]       | CWG2B_out         |
| 110001 [49]       | CWG2A_out         |
| 110000 [48]       | CWG1B_out         |
| 101111 [47]       | CWG1A_out         |
| 101110 [46]       | SS1               |
| 101101 [45]       | SCK1              |
| 101100 [44]       | SDO1              |
| 101011 [43]       | Reserved          |
| 101010 [42]       | UART2_tx_out      |
| 101001 [41]       | UART1_tx_out      |
| 101000 [40]       | CLC4_out          |
| 100111 [39]       | CLC3_out          |
| 100110 [38]       | CLC2_out          |
| 100101 [37]       | CLC1_out          |
| 100100 [36]       | DSM1_out          |
| 100011 [35]       | IOC_flag          |
| 100010 [34]       | ZCD_out           |
| 100001 [33]       | CMP2_out          |
| 100000 [32]       | CMP1_out          |
| 011111 [31]       | NCO1_out          |
| 011110 [30]       | Reserved          |
| 011101 [29]       | Reserved          |
| 011100 [28]       | PWM8_out          |
| 011011 [27]       | PWM7_out          |
| 011010 [26]       | PWM6_out          |
| 011001 [25]       | PWM5_out          |
| 011000 [24]       | CCP4_out          |
| 010111 [23]       | CCP3_out          |
| 010110 [22]       | CCP2_out          |
| 010101 [21]       | CCP1_out          |
| 010100 [20]       | SMT2_out          |
| 010011 [19]       | SMT1_out          |
| 010010 [18]       | TMR6_out          |

**TABLE 27-1: CLCx DATA INPUT SELECTION (CONTINUED)**

| DyS<5:0><br>Value | CLCx Input Source  |
|-------------------|--------------------|
| 010001 [17]       | TMR5_overflow      |
| 010000 [16]       | TMR4_out           |
| 001111 [15]       | TMR3_overflow      |
| 001110 [14]       | TMR2_out           |
| 001101 [13]       | TMR1_overflow      |
| 001100 [12]       | TMR0_overflow      |
| 001011 [11]       | CLKR_out           |
| 001010 [10]       | ADCRC              |
| 001001 [9]        | SOSC               |
| 001000 [8]        | MFINTOSC (32 kHz)  |
| 000111 [7]        | MFINTOSC (500 kHz) |
| 000110 [6]        | LFINTOSC           |
| 000101 [5]        | HFINTOSC           |
| 000100 [4]        | Fosc               |
| 000011 [3]        | CLCIN3PPS          |
| 000010 [2]        | CLCIN2PPS          |
| 000001 [1]        | CLCIN1PPS          |
| 000000 [0]        | CLCIN0PPS          |

## 27.1.2 DATA GATING

Outputs from the input multiplexers are directed to the desired logic function input through the data gating stage. Each data gate can direct any combination of the four selected inputs.

**Note:** Data gating is undefined at power-up.

The gate stage is more than just signal direction. The gate can be configured to direct each input signal as inverted or non-inverted data. Directed signals are ANDed together in each gate. The output of each gate can be inverted before going on to the logic function stage.

The gating is in essence a 1-to-4 input AND/NAND/OR/NOR gate. When every input is inverted and the output is inverted, the gate is an OR of all enabled data inputs. When the inputs and output are not inverted, the gate is an AND of all enabled inputs.

Table 27-2 summarizes the basic logic that can be obtained in gate 1 by using the gate logic select bits. The table shows the logic of four input variables, but each gate can be configured to use less than four. If no inputs are selected, the output will be zero or one, depending on the gate output polarity bit.

**TABLE 27-2: DATA GATING LOGIC**

| CLCxGLSy | GyPOL | Gate Logic |
|----------|-------|------------|
| 0x55     | 1     | AND        |
| 0x55     | 0     | NAND       |
| 0xAA     | 1     | NOR        |
| 0xAA     | 0     | OR         |
| 0x00     | 0     | Logic 0    |
| 0x00     | 1     | Logic 1    |

It is possible (but not recommended) to select both the true and negated values of an input. When this is done, the gate output is zero, regardless of the other inputs, but may emit logic glitches (transient-induced pulses). If the output of the channel must be zero or one, the recommended method is to set all gate bits to zero and use the gate polarity bit to set the desired level.

Data gating is configured with the logic gate select registers as follows:

- Gate 1: CLCxGLS0 (Register 27-7)
- Gate 2: CLCxGLS1 (Register 27-8)
- Gate 3: CLCxGLS2 (Register 27-9)
- Gate 4: CLCxGLS3 (Register 27-10)

Register number suffixes are different than the gate numbers because other variations of this module have multiple gate selections in the same register.

Data gating is indicated in the right side of Figure 27-2. Only one gate is shown in detail. The remaining three gates are configured identically with the exception that the data enables correspond to the enables for that gate.

## 27.1.3 LOGIC FUNCTION

There are eight available logic functions including:

- AND-OR
- OR-XOR
- AND
- S-R Latch
- D Flip-Flop with Set and Reset
- D Flip-Flop with Reset
- J-K Flip-Flop with Reset
- Transparent Latch with Set and Reset

Logic functions are shown in Figure 27-2. Each logic function has four inputs and one output. The four inputs are the four data gate outputs of the previous stage. The output is fed to the inversion stage and from there to other peripherals, an output pin, and back to the CLCx itself.

## 27.1.4 OUTPUT POLARITY

The last stage in the Configurable Logic Cell is the output polarity. Setting the POL bit of the CLCxPOL register inverts the output signal from the logic stage. Changing the polarity while the interrupts are enabled will cause an interrupt for the resulting output transition.

## 27.2 CLCx Interrupts

An interrupt will be generated upon a change in the output value of the CLCx when the appropriate interrupt enables are set. A rising edge detector and a falling edge detector are present in each CLC for this purpose.

The CLCxIF bit of the associated PIR5 register will be set when either edge detector is triggered and its associated enable bit is set. The INTP enables rising edge interrupts and the INTN bit enables falling edge interrupts. Both are located in the CLCxCON register.

To fully enable the interrupt, set the following bits:

- CLCxIE bit of the respective PIE register
- INTP bit of the CLCxCON register (for a rising edge detection)
- INTN bit of the CLCxCON register (for a falling edge detection)
- GIE bits of the INTCON0 register

The CLCxIF bit of the respective PIR register, must be cleared in software as part of the interrupt service. If another edge is detected while this flag is being cleared, the flag will still be set at the end of the sequence.

## 27.3 Output Mirror Copies

Mirror copies of all CON output bits are contained in the CLCxDATA register. Reading this register reads the outputs of all CLCs simultaneously. This prevents any reading skew introduced by testing or reading the OUT bits in the individual CLCxCON registers.

## 27.4 Effects of a Reset

The CLCxCON register is cleared to zero as the result of a Reset. All other selection and gating values remain unchanged.

## 27.5 Operation During Sleep

The CLC module operates independently from the system clock and will continue to run during Sleep, provided that the input sources selected remain active.

The HFINTOSC remains active during Sleep when the CLC module is enabled and the HFINTOSC is selected as an input source, regardless of the system clock source selected.

In other words, if the HFINTOSC is simultaneously selected as the system clock and as a CLC input source, when the CLC is enabled, the CPU will go Idle during Sleep, but the CLC will continue to operate and the HFINTOSC will remain active.

This will have a direct effect on the Sleep mode current.

## 27.6 CLCx Setup Steps

The following steps should be followed when setting up the CLCx:

- Disable CLCx by clearing the EN bit.
- Select desired inputs using CLCxSEL0 through CLCxSEL3 registers (See [Table 27-1](#)).
- Clear any associated ANSEL bits.
- Set all TRIS bits associated with inputs.
- Clear all TRIS bits associated with outputs.
- Enable the chosen inputs through the four gates using CLCxGLS0, CLCxGLS1, CLCxGLS2, and CLCxGLS3 registers.
- Select the gate output polarities with the GypOL bits of the CLCxPOL register.
- Select the desired logic function with the MODE<2:0> bits of the CLCxCON register.
- Select the desired polarity of the logic output with the POL bit of the CLCxPOL register. (This step may be combined with the previous gate output polarity step).
- If driving a device pin, set the desired pin PPS control register and also clear the TRIS bit corresponding to that output.
- If interrupts are desired, configure the following bits:
  - Set the INTP bit in the CLCxCON register for rising event.
  - Set the INTN bit in the CLCxCON register for falling event.
  - Set the CLCxIE bit of the respective PIE register.
  - Set the GIE bits of the INTCON0 register.
- Enable the CLCx by setting the EN bit of the CLCxCON register.



**FIGURE 27-2: INPUT DATA SELECTION AND GATING**

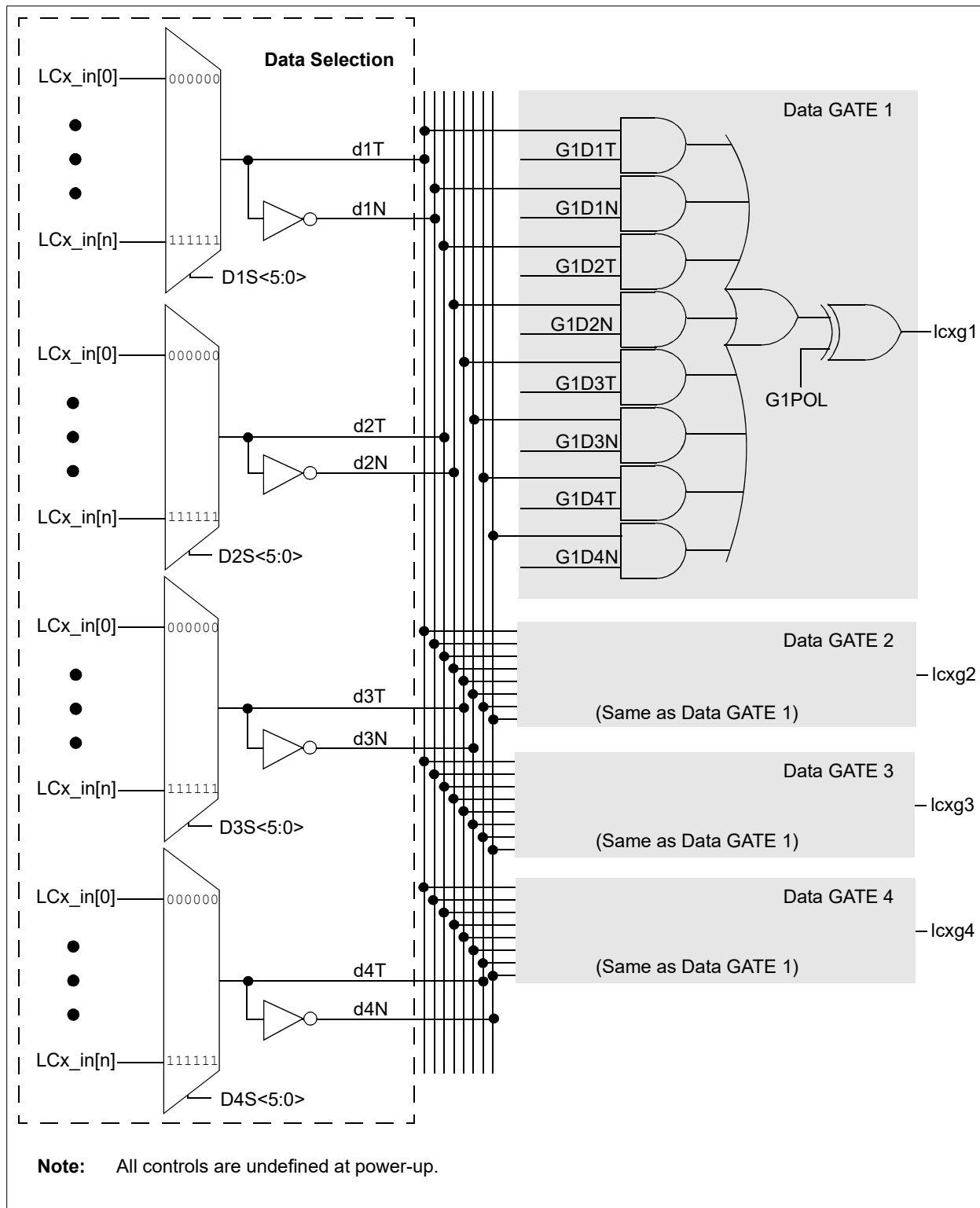
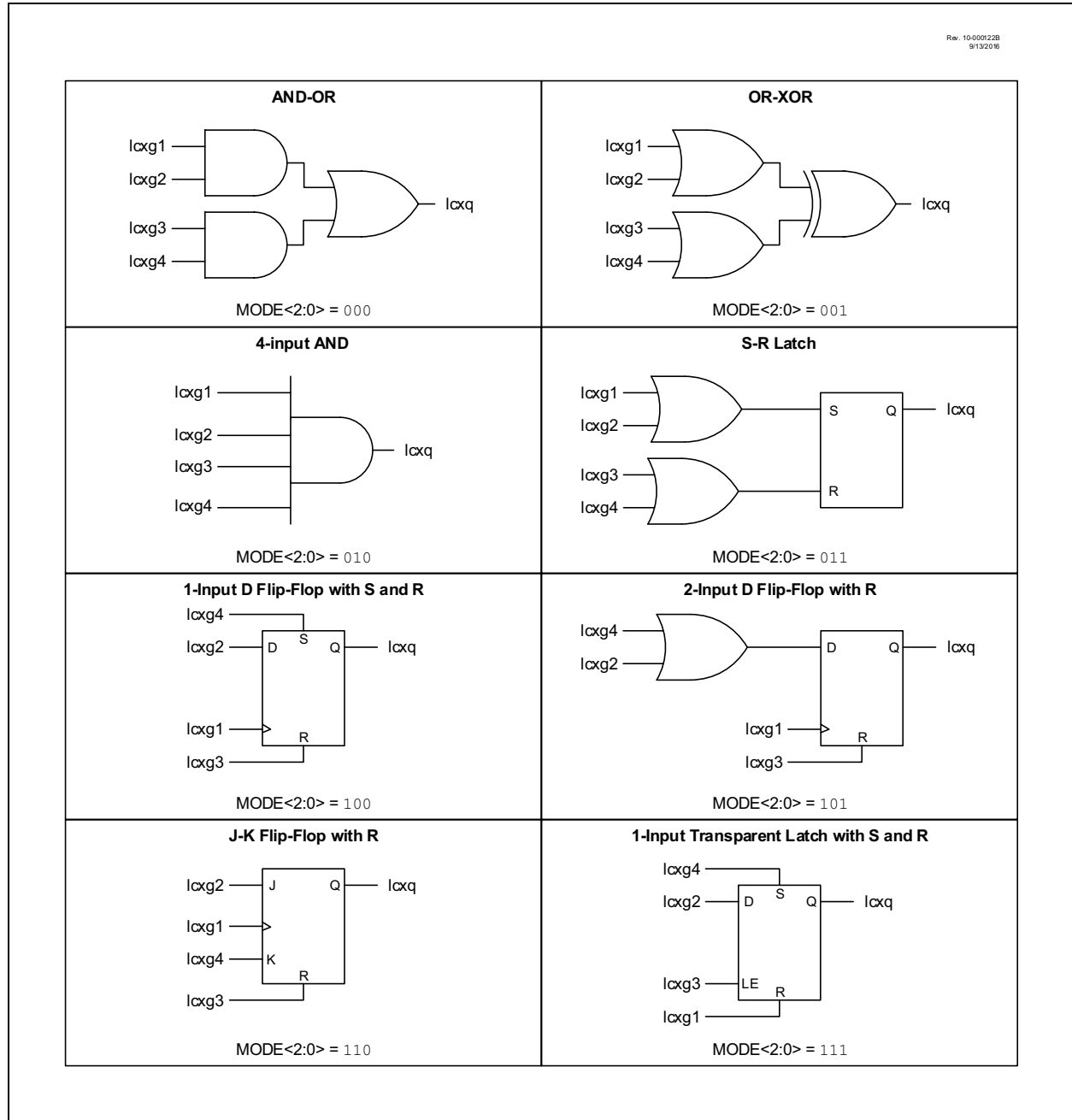


**FIGURE 27-3: PROGRAMMABLE LOGIC FUNCTIONS**



## 27.7 Register Definitions: CLC Control

**REGISTER 27-1: CLCxCON: CONFIGURABLE LOGIC CELL CONTROL REGISTER**

| R/W-0/0 | U-0 | R-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0   | R/W-0/0 | R/W-0/0 |
|---------|-----|-------|---------|---------|-----------|---------|---------|
| EN      | —   | OUT   | INTP    | INTN    | MODE<2:0> |         |         |
| bit 7   |     |       |         |         |           |         | bit 0   |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

- bit 7      **EN:** Configurable Logic Cell Enable bit  
1 = Configurable logic cell is enabled and mixing input signals  
0 = Configurable logic cell is disabled and has logic zero output
- bit 6      **Unimplemented:** Read as '0'
- bit 5      **OUT:** Configurable Logic Cell Data Output bit  
Read-only: logic cell output data, after LCPOL; sampled from CLCxOUT
- bit 4      **INTP:** Configurable Logic Cell Positive Edge Going Interrupt Enable bit  
1 = CLCxIF will be set when a rising edge occurs on CLCxOUT  
0 = CLCxIF will not be set
- bit 3      **INTN:** Configurable Logic Cell Negative Edge Going Interrupt Enable bit  
1 = CLCxIF will be set when a falling edge occurs on CLCxOUT  
0 = CLCxIF will not be set
- bit 2-0    **MODE<2:0>:** Configurable Logic Cell Functional Mode bits  
111 = Cell is 1-input transparent latch with S and R  
110 = Cell is J-K flip-flop with R  
101 = Cell is 2-input D flip-flop with R  
100 = Cell is 1-input D flip-flop with S and R  
011 = Cell is S-R latch  
010 = Cell is 4-input AND  
001 = Cell is OR-XOR  
000 = Cell is AND-OR

## REGISTER 27-2: CLCxPOL: SIGNAL POLARITY CONTROL REGISTER

| R/W-0/0 | U-0 | U-0 | U-0 | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
|---------|-----|-----|-----|---------|---------|---------|---------|
| POL     | —   | —   | —   | G4POL   | G3POL   | G2POL   | G1POL   |
| bit 7   |     |     |     |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

- bit 7      **POL:** CLCxOUT Output Polarity Control bit  
1 = The output of the logic cell is inverted  
0 = The output of the logic cell is not inverted
- bit 6-4    **Unimplemented:** Read as '0'
- bit 3      **G4POL:** Gate 3 Output Polarity Control bit  
1 = The output of gate 3 is inverted when applied to the logic cell  
0 = The output of gate 3 is not inverted
- bit 2      **G3POL:** Gate 2 Output Polarity Control bit  
1 = The output of gate 2 is inverted when applied to the logic cell  
0 = The output of gate 2 is not inverted
- bit 1      **G2POL:** Gate 1 Output Polarity Control bit  
1 = The output of gate 1 is inverted when applied to the logic cell  
0 = The output of gate 1 is not inverted
- bit 0      **G1POL:** Gate 0 Output Polarity Control bit  
1 = The output of gate 0 is inverted when applied to the logic cell  
0 = The output of gate 0 is not inverted

## REGISTER 27-3: CLCxSEL0: GENERIC CLCx DATA 0 SELECT REGISTER

| U-0   | U-0 | R/W-x/u  | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
|-------|-----|----------|---------|---------|---------|---------|---------|
| —     | —   | D1S<5:0> |         |         |         |         |         |
| bit 7 |     |          |         |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-6

**Unimplemented:** Read as '0'

bit 5-0

**D1S<5:0>**: CLCx Data1 Input Selection bits  
See [Table 27-1](#).

## REGISTER 27-4: CLCxSEL1: GENERIC CLCx DATA 1 SELECT REGISTER

| U-0   | U-0 | R/W-x/u  | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
|-------|-----|----------|---------|---------|---------|---------|---------|
| —     | —   | D2S<5:0> |         |         |         |         |         |
| bit 7 |     |          |         |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-6

**Unimplemented:** Read as '0'

bit 5-0

**D2S<5:0>**: CLCx Data 2 Input Selection bits  
See [Table 27-1](#).

## REGISTER 27-5: CLCxSEL2: GENERIC CLCx DATA 2 SELECT REGISTER

| U-0   | U-0 | R/W-x/u  | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
|-------|-----|----------|---------|---------|---------|---------|---------|
| —     | —   | D3S<5:0> |         |         |         |         |         |
| bit 7 |     |          |         |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-6

**Unimplemented:** Read as '0'

bit 5-0

**D3S<5:0>**: CLCx Data 3 Input Selection bits  
See [Table 27-1](#).

## REGISTER 27-6: CLCxSEL3: GENERIC CLCx DATA 3 SELECT REGISTER

| U-0   | U-0 | R/W-x/u  | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
|-------|-----|----------|---------|---------|---------|---------|---------|
| —     | —   | D4S<5:0> |         |         |         |         |         |
| bit 7 |     |          |         |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-6

**Unimplemented:** Read as '0'

bit 5-0

**D4S<5:0>**: CLCx Data 4 Input Selection bits  
See [Table 27-1](#).

## REGISTER 27-7: CLCxGLS0: GATE 0 LOGIC SELECT REGISTER

| R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
|---------|---------|---------|---------|---------|---------|---------|---------|
| G1D4T   | G1D4N   | G1D3T   | G1D3N   | G1D2T   | G1D2N   | G1D1T   | G1D1N   |
| bit 7   |         |         |         |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

- bit 7      **G1D4T:** Gate 0 Data 4 True (non-inverted) bit  
1 = CLCIN3 (true) is gated into CLCx Gate 0  
0 = CLCIN3 (true) is not gated into CLCx Gate 0
- bit 6      **G1D4N:** Gate 0 Data 4 Negated (inverted) bit  
1 = CLCIN3 (inverted) is gated into CLCx Gate 0  
0 = CLCIN3 (inverted) is not gated into CLCx Gate 0
- bit 5      **G1D3T:** Gate 0 Data 3 True (non-inverted) bit  
1 = CLCIN2 (true) is gated into CLCx Gate 0  
0 = CLCIN2 (true) is not gated into CLCx Gate 0
- bit 4      **G1D3N:** Gate 0 Data 3 Negated (inverted) bit  
1 = CLCIN2 (inverted) is gated into CLCx Gate 0  
0 = CLCIN2 (inverted) is not gated into CLCx Gate 0
- bit 3      **G1D2T:** Gate 0 Data 2 True (non-inverted) bit  
1 = CLCIN1 (true) is gated into CLCx Gate 0  
0 = CLCIN1 (true) is not gated into CLCx Gate 0
- bit 2      **G1D2N:** Gate 0 Data 2 Negated (inverted) bit  
1 = CLCIN1 (inverted) is gated into CLCx Gate 0  
0 = CLCIN1 (inverted) is not gated into CLCx Gate 0
- bit 1      **G1D1T:** Gate 0 Data 1 True (non-inverted) bit  
1 = CLCIN0 (true) is gated into CLCx Gate 0  
0 = CLCIN0 (true) is not gated into CLCx Gate 0
- bit 0      **G1D1N:** Gate 0 Data 1 Negated (inverted) bit  
1 = CLCIN0 (inverted) is gated into CLCx Gate 0  
0 = CLCIN0 (inverted) is not gated into CLCx Gate 0

## REGISTER 27-8: CLCxGLS1: GATE 1 LOGIC SELECT REGISTER

| R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
|---------|---------|---------|---------|---------|---------|---------|---------|
| G2D4T   | G2D4N   | G2D3T   | G2D3N   | G2D2T   | G2D2N   | G2D1T   | G2D1N   |
| bit 7   |         |         |         |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

- bit 7      **G2D4T:** Gate 1 Data 4 True (noninverted) bit  
1 = CLCIN3 (true) is gated into CLCx Gate 1  
0 = CLCIN3 (true) is not gated into CLCx Gate 1
- bit 6      **G2D4N:** Gate 1 Data 4 Negated (inverted) bit  
1 = CLCIN3 (inverted) is gated into CLCx Gate 1  
0 = CLCIN3 (inverted) is not gated into CLCx Gate 1
- bit 5      **G2D3T:** Gate 1 Data 3 True (noninverted) bit  
1 = CLCIN2 (true) is gated into CLCx Gate 1  
0 = CLCIN2 (true) is not gated into CLCx Gate 1
- bit 4      **G2D3N:** Gate 1 Data 3 Negated (inverted) bit  
1 = CLCIN2 (inverted) is gated into CLCx Gate 1  
0 = CLCIN2 (inverted) is not gated into CLCx Gate 1
- bit 3      **G2D2T:** Gate 1 Data 2 True (noninverted) bit  
1 = CLCIN1 (true) is gated into CLCx Gate 1  
0 = CLCIN1 (true) is not gated into CLCx Gate 1
- bit 2      **G2D2N:** Gate 1 Data 2 Negated (inverted) bit  
1 = CLCIN1 (inverted) is gated into CLCx Gate 1  
0 = CLCIN1 (inverted) is not gated into CLCx Gate 1
- bit 1      **G2D1T:** Gate 1 Data 1 True (noninverted) bit  
1 = CLCIN0 (true) is gated into CLCx Gate 1  
0 = CLCIN0 (true) is not gated into CLCx Gate 1
- bit 0      **G2D1N:** Gate 1 Data 1 Negated (inverted) bit  
1 = CLCIN0 (inverted) is gated into CLCx Gate 1  
0 = CLCIN0 (inverted) is not gated into CLCx Gate 1

## REGISTER 27-9: CLCxGLS2: GATE 2 LOGIC SELECT REGISTER

| R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
|---------|---------|---------|---------|---------|---------|---------|---------|
| G3D4T   | G3D4N   | G3D3T   | G3D3N   | G3D2T   | G3D2N   | G3D1T   | G3D1N   |
| bit 7   |         |         |         |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

- bit 7      **G3D4T:** Gate 2 Data 4 True (noninverted) bit  
1 = CLCIN3 (true) is gated into CLCx Gate 2  
0 = CLCIN3 (true) is not gated into CLCx Gate 2
- bit 6      **G3D4N:** Gate 2 Data 4 Negated (inverted) bit  
1 = CLCIN3 (inverted) is gated into CLCx Gate 2  
0 = CLCIN3 (inverted) is not gated into CLCx Gate 2
- bit 5      **G3D3T:** Gate 2 Data 3 True (noninverted) bit  
1 = CLCIN2 (true) is gated into CLCx Gate 2  
0 = CLCIN2 (true) is not gated into CLCx Gate 2
- bit 4      **G3D3N:** Gate 2 Data 3 Negated (inverted) bit  
1 = CLCIN2 (inverted) is gated into CLCx Gate 2  
0 = CLCIN2 (inverted) is not gated into CLCx Gate 2
- bit 3      **G3D2T:** Gate 2 Data 2 True (noninverted) bit  
1 = CLCIN1 (true) is gated into CLCx Gate 2  
0 = CLCIN1 (true) is not gated into CLCx Gate 2
- bit 2      **G3D2N:** Gate 2 Data 2 Negated (inverted) bit  
1 = CLCIN1 (inverted) is gated into CLCx Gate 2  
0 = CLCIN1 (inverted) is not gated into CLCx Gate 2
- bit 1      **G3D1T:** Gate 2 Data 1 True (noninverted) bit  
1 = CLCIN0 (true) is gated into CLCx Gate 2  
0 = CLCIN0 (true) is not gated into CLCx Gate 2
- bit 0      **G3D1N:** Gate 2 Data 1 Negated (inverted) bit  
1 = CLCIN0 (inverted) is gated into CLCx Gate 2  
0 = CLCIN0 (inverted) is not gated into CLCx Gate 2



## REGISTER 27-10: CLCxGLS3: GATE 3 LOGIC SELECT REGISTER

| R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
|---------|---------|---------|---------|---------|---------|---------|---------|
| G4D4T   | G4D4N   | G4D3T   | G4D3N   | G4D2T   | G4D2N   | G4D1T   | G4D1N   |
| bit 7   |         |         |         |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

- bit 7      **G4D4T:** Gate 3 Data 4 True (non-inverted) bit  
1 = CLCIN3 (true) is gated into CLCx Gate 3  
0 = CLCIN3 (true) is not gated into CLCx Gate 3
- bit 6      **G4D4N:** Gate 3 Data 4 Negated (inverted) bit  
1 = CLCIN3 (inverted) is gated into CLCx Gate 3  
0 = CLCIN3 (inverted) is not gated into CLCx Gate 3
- bit 5      **G4D3T:** Gate 3 Data 3 True (non-inverted) bit  
1 = CLCIN2 (true) is gated into CLCx Gate 3  
0 = CLCIN2 (true) is not gated into CLCx Gate 3
- bit 4      **G4D3N:** Gate 3 Data 3 Negated (inverted) bit  
1 = CLCIN2 (inverted) is gated into CLCx Gate 3  
0 = CLCIN2 (inverted) is not gated into CLCx Gate 3
- bit 3      **G4D2T:** Gate 3 Data 2 True (non-inverted) bit  
1 = CLCIN1 (true) is gated into CLCx Gate 3  
0 = CLCIN1 (true) is not gated into CLCx Gate 3
- bit 2      **G4D2N:** Gate 3 Data 2 Negated (inverted) bit  
1 = CLCIN1 (inverted) is gated into CLCx Gate 3  
0 = CLCIN1 (inverted) is not gated into CLCx Gate 3
- bit 1      **G4D1T:** Gate 4 Data 1 True (non-inverted) bit  
1 = CLCIN0 (true) is gated into CLCx Gate 3  
0 = CLCIN0 (true) is not gated into CLCx Gate 3
- bit 0      **G4D1N:** Gate 3 Data 1 Negated (inverted) bit  
1 = CLCIN0 (inverted) is gated into CLCx Gate 3  
0 = CLCIN0 (inverted) is not gated into CLCx Gate 3

## REGISTER 27-11: CLCDATA: CLC DATA OUTPUT

|       |     |     |     |         |         |         |         |
|-------|-----|-----|-----|---------|---------|---------|---------|
| U-0   | U-0 | U-0 | U-0 | R-0     | R-0     | R-0     | R-0     |
| —     | —   | —   | —   | CLC4OUT | CLC3OUT | CLC2OUT | CLC1OUT |
| bit 7 |     |     |     |         |         |         | bit 0   |

### Legend:

R = Readable bit                      W = Writable bit                      U = Unimplemented bit, read as '0'  
 u = Bit is unchanged                  x = Bit is unknown                  -n/n = Value at POR and BOR/Value at all other Resets  
 '1' = Bit is set                          '0' = Bit is cleared

bit 7-4                      **Unimplemented:** Read as '0'  
 bit 3                      **CLC4OUT:** Mirror copy of OUT bit of CLC4CON register  
 bit 2                      **CLC3OUT:** Mirror copy of OUT bit of CLC3CON register  
 bit 1                      **CLC2OUT:** Mirror copy of OUT bit of CLC2CON register  
 bit 0                      **CLC1OUT:** Mirror copy of OUT bit of CLC1CON register

**TABLE 27-3: SUMMARY OF REGISTERS ASSOCIATED WITH CLCx**

| Name     | Bit 7 | Bit 6 | Bit 5    | Bit 4 | Bit 3   | Bit 2     | Bit 1   | Bit 0   | Register on Page |
|----------|-------|-------|----------|-------|---------|-----------|---------|---------|------------------|
| CLCxCON  | EN    | —     | OUT      | INTP  | INTN    | MODE<2:0> |         |         | <Blue>427        |
| CLCxPOL  | POL   | —     | —        | —     | G4POL   | G3POL     | G2POL   | G1POL   | <Blue>428        |
| CLCxSEL0 | —     | —     | D1S<5:0> |       |         |           |         |         | <Blue>429        |
| CLCxSEL1 | —     | —     | D2S<5:0> |       |         |           |         |         | <Blue>429        |
| CLCxSEL2 | —     | —     | D3S<5:0> |       |         |           |         |         | <Blue>429        |
| CLCxSEL3 | —     | —     | D4S<5:0> |       |         |           |         |         | <Blue>429        |
| CLCxGLS0 | G1D4T | G1D4N | G1D3T    | G1D3N | G1D2T   | G1D2N     | G1D1T   | G1D1N   | <Blue>430        |
| CLCxGLS1 | G2D4T | G2D4N | G2D3T    | G2D3N | G2D2T   | G2D2N     | G2D1T   | G2D1N   | <Blue>431        |
| CLCxGLS2 | G3D4T | G3D4N | G3D3T    | G3D3N | G3D2T   | G3D2N     | G3D1T   | G3D1N   | <Blue>432        |
| CLCxGLS3 | G4D4T | G4D4N | G4D3T    | G4D3N | G4D2T   | G4D2N     | G4D1T   | G4D1N   | <Blue>433        |
| CLCDATA  | —     | —     | —        | —     | CLC4OUT | CLC3OUT   | CLC2OUT | CLC1OUT | <Blue>434        |

**Legend:** — = unimplemented, read as '0'. Shaded cells are unused by the CLCx modules.

## 28.0 NUMERICALLY CONTROLLED OSCILLATOR (NCO) MODULE

The Numerically Controlled Oscillator (NCO) module is a timer that uses overflow from the addition of an increment value to divide the input frequency. The advantage of the addition method over simple counter driven timer is that the output frequency resolution does not vary with the divider value. The NCO is most useful for application that requires frequency accuracy and fine resolution at a fixed duty cycle.

Features of the NCO include:

- 20-bit Increment Function
- Fixed Duty Cycle mode (FDC) mode
- Pulse Frequency (PF) mode
- Output Pulse-Width Control
- Multiple Clock Input Sources
- Output Polarity Control
- Interrupt Capability

Figure 28-1 is a simplified block diagram of the NCO module.



## 28.1 NCO Operation

The NCO operates by repeatedly adding a fixed value to an accumulator. Additions occur at the input clock rate. The accumulator will overflow with a carry periodically, which is the raw NCO output (NCO\_overflow). This effectively reduces the input clock by the ratio of the addition value to the maximum accumulator value. See [Equation 28-1](#).

The NCO output can be further modified by stretching the pulse or toggling a flip-flop. The modified NCO output is then distributed internally to other peripherals and can be optionally output to a pin. The accumulator overflow also generates an interrupt (NCO\_overflow).

The NCO period changes in discrete steps to create an average frequency. This output depends on the ability of the receiving circuit (i.e., CWG or external resonant converter circuitry) to average the NCO output to reduce uncertainty.

### EQUATION 28-1: NCO OVERFLOW FREQUENCY

$$F_{\text{OVERFLOW}} = \frac{\text{NCO Clock Frequency} \times \text{Increment Value}}{2^{20}}$$

#### 28.1.1 NCO CLOCK SOURCES

Clock sources available to the NCO include:

- FOSC
- HFINTOSC
- LFINTOSC
- MFINTOSC/4 (32 kHz)
- MFINTOSC (500 kHz)
- CLC1/2/3/4\_out
- CLKREF
- SOSC

The NCO clock source is selected by configuring the N1CKS<2:0> bits in the NCO1CLK register.

#### 28.1.2 ACCUMULATOR

The accumulator is a 20-bit register. Read and write access to the accumulator is available through three registers:

- NCO1ACCL
- NCO1ACCH
- NCO1ACCU

#### 28.1.3 ADDER

The NCO Adder is a full adder, which operates independently from the source clock. The addition of the previous result and the increment value replaces the accumulator value on the rising edge of each input clock.

#### 28.1.4 INCREMENT REGISTERS

The increment value is stored in three registers making up a 20-bit incrementer. In order of LSB to MSB they are:

- NCO1INCL
- NCO1INCH
- NCO1INCUI

When the NCO module is enabled, the NCO1INCUI and NCO1INCH registers should be written first, then the NCO1INCL register. Writing to the NCO1INCL register initiates the increment buffer registers to be loaded simultaneously on the second rising edge of the NCO\_clk signal.

The registers are readable and writable. The increment registers are double-buffered to allow value changes to be made without first disabling the NCO module.

When the NCO module is disabled, the increment buffers are loaded immediately after a write to the increment registers.

**Note:** The increment buffer registers are not user-accessible.

## 28.2 FIXED DUTY CYCLE MODE

In Fixed Duty Cycle (FDC) mode, every time the accumulator overflows (NCO\_overflow), the output is toggled. This provides a 50% duty cycle, provided that the increment value remains constant. For more information, see [Figure 28-2](#).

## 28.3 PULSE FREQUENCY MODE

In Pulse Frequency (PF) mode, every time the Accumulator overflows, the output becomes active for one or more clock periods. Once the clock period expires, the output returns to an inactive state. This provides a pulsed output. The output becomes active on the rising clock edge immediately following the overflow event. For more information, see [Figure 28-2](#).

The value of the active and inactive states depends on the polarity bit, POL in the NCO1CON register.

The PF mode is selected by setting the PFM bit in the NCO1CON register.

### 28.3.1 OUTPUT PULSE-WIDTH CONTROL

When operating in PF mode, the active state of the output can vary in width by multiple clock periods. Various pulse widths are selected with the PWS<2:0> bits in the NCO1CLK register.

When the selected pulse width is greater than the Accumulator overflow time frame, then DDS operation is undefined.

## 28.4 OUTPUT POLARITY CONTROL

The last stage in the NCO module is the output polarity. The POL bit in the NCO1CON register selects the output polarity. Changing the polarity while the interrupts are enabled will cause an interrupt for the resulting output transition. The NCO output signal is available to most of the other peripherals available on the device.

## 28.5 Interrupts

When the accumulator overflows (NCO\_overflow), the NCO Interrupt Flag bit, NCO1IF, of the PIR4 register is set. To enable the interrupt event (NCO\_interrupt), the following bits must be set:

- EN bit of the NCO1CON register
- NCO1IE bit of the PIE4 register
- GIE/GIEH bit of the INTCON0 register

The interrupt must be cleared by software by clearing the NCO1IF bit in the Interrupt Service Routine.

## 28.6 Effects of a Reset

All of the NCO registers are cleared to zero as the result of a Reset.

## 28.7 Operation in Sleep

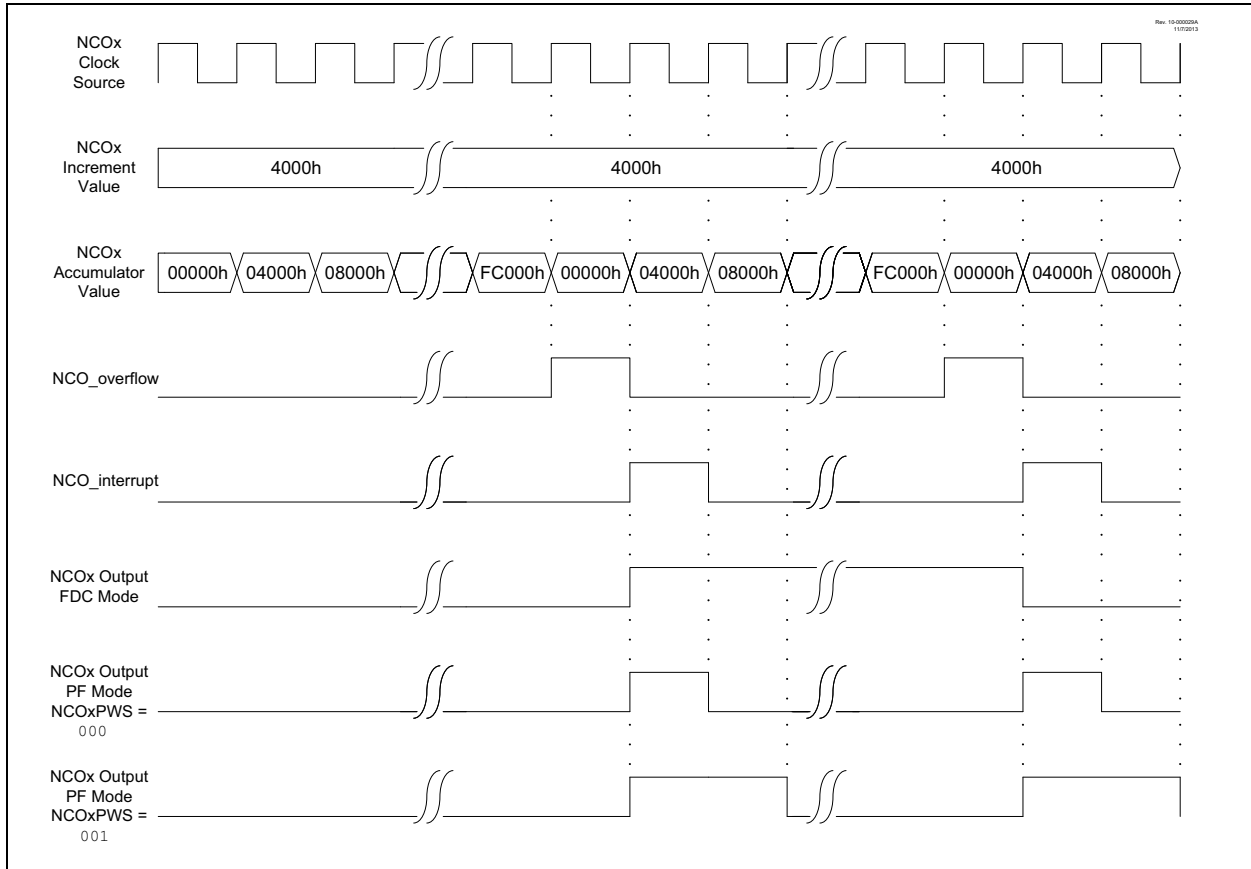
The NCO module operates independently from the system clock and will continue to run during Sleep, provided that the clock source selected remains active.

The HFINTOSC remains active during Sleep when the NCO module is enabled and the HFINTOSC is selected as the clock source, regardless of the system clock source selected.

In other words, if the HFINTOSC is simultaneously selected as the system clock and the NCO clock source, when the NCO is enabled, the CPU will go Idle during Sleep, but the NCO will continue to operate and the HFINTOSC will remain active.

This will have a direct effect on the Sleep mode current.

**FIGURE 28-2: FDC OUTPUT MODE OPERATION DIAGRAM**



## 28.8 NCO Control Registers

**REGISTER 28-1: NCO1CON: NCO CONTROL REGISTER**

| R/W-0/0 | U-0 | R-0/0 | R/W-0/0 | U-0 | U-0 | U-0 | R/W-0/0 |
|---------|-----|-------|---------|-----|-----|-----|---------|
| EN      | —   | OUT   | POL     | —   | —   | —   | PFM     |
| bit 7   |     |       |         |     |     |     | bit 0   |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

- bit 7      **EN:** NCO1 Enable bit  
1 = NCO1 module is enabled  
0 = NCO1 module is disabled
- bit 6      **Unimplemented:** Read as '0'
- bit 5      **OUT:** NCO1 Output bit  
Displays the current output value of the NCO1 module.
- bit 4      **POL:** NCO1 Polarity  
1 = NCO1 output signal is inverted  
0 = NCO1 output signal is not inverted
- bit 3-1    **Unimplemented:** Read as '0'
- bit 0      **PFM:** NCO1 Pulse Frequency Mode bit  
1 = NCO1 operates in Pulse Frequency mode  
0 = NCO1 operates in Fixed Duty Cycle mode, divide by 2



**REGISTER 28-2: NCO1CLK: NCO1 INPUT CLOCK CONTROL REGISTER**

|                           |         |         |     |          |         |         |         |
|---------------------------|---------|---------|-----|----------|---------|---------|---------|
| R/W-0/0                   | R/W-0/0 | R/W-0/0 | U-0 | R/W-0/0  | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| PWS<2:0> <sup>(1,2)</sup> |         |         | —   | CKS<3:0> |         |         |         |
| bit 7                     |         |         |     | bit 0    |         |         |         |

**Legend:**

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-5      **PWS<2:0>**: NCO1 Output Pulse Width Select bits<sup>(1,2)</sup>

- 111 = NCO1 output is active for 128 input clock periods
- 110 = NCO1 output is active for 64 input clock periods
- 101 = NCO1 output is active for 32 input clock periods
- 100 = NCO1 output is active for 16 input clock periods
- 011 = NCO1 output is active for 8 input clock periods
- 010 = NCO1 output is active for 4 input clock periods
- 001 = NCO1 output is active for 2 input clock periods
- 000 = NCO1 output is active for 1 input clock period

bit 4      **Unimplemented**: Read as '0'

bit 3-0      **CKS<3:0>**: NCO1 Clock Source Select bits

- 1111 = Reserved
- 
- 
- 
- 1011 = Reserved
- 1010 = CLC4\_out
- 1001 = CLC3\_out
- 1000 = CLC2\_out
- 0111 = CLC1\_out
- 0110 = CLKREF\_out
- 0101 = SOSC
- 0100 = MFINTOSC/4 (32 kHz)
- 0011 = MFINTOSC (500 kHz)
- 0010 = LFINTOSC
- 0001 = HFINTOSC
- 0000 = Fosc

**Note 1:** N1PWS applies only when operating in Pulse Frequency mode.

**2:** If NCO1 pulse width is greater than NCO1 overflow period, operation is undefined.

## REGISTER 28-3: NCO1ACCL: NCO1 ACCUMULATOR REGISTER – LOW BYTE

|          |         |         |         |         |         |         |         |
|----------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0  | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| ACC<7:0> |         |         |         |         |         |         |         |
| bit 7    |         |         |         | bit 0   |         |         |         |

### Legend:

R = Readable bit                      W = Writable bit                      U = Unimplemented bit, read as '0'  
 u = Bit is unchanged                  x = Bit is unknown                      -n/n = Value at POR and BOR/Value at all other Resets  
 '1' = Bit is set                          '0' = Bit is cleared

bit 7-0                  **ACC<7:0>**: NCO1 Accumulator, Low Byte

## REGISTER 28-4: NCO1ACCH: NCO1 ACCUMULATOR REGISTER – HIGH BYTE

|           |         |         |         |         |         |         |         |
|-----------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0   | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| ACC<15:8> |         |         |         |         |         |         |         |
| bit 7     |         |         |         | bit 0   |         |         |         |

### Legend:

R = Readable bit                      W = Writable bit                      U = Unimplemented bit, read as '0'  
 u = Bit is unchanged                  x = Bit is unknown                      -n/n = Value at POR and BOR/Value at all other Resets  
 '1' = Bit is set                          '0' = Bit is cleared

bit 7-0                  **ACC<15:8>**: NCO1 Accumulator, High Byte

## REGISTER 28-5: NCO1ACCU: NCO1 ACCUMULATOR REGISTER – UPPER BYTE<sup>(1)</sup>

|       |     |     |     |            |         |         |         |
|-------|-----|-----|-----|------------|---------|---------|---------|
| U-0   | U-0 | U-0 | U-0 | R/W-0/0    | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| —     | —   | —   | —   | ACC<19:16> |         |         |         |
| bit 7 |     |     |     | bit 0      |         |         |         |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-4 **Unimplemented:** Read as '0'

bit 3-0 **ACC<19:16>:** NCO1 Accumulator, Upper Byte

**Note 1:** The accumulator spans registers NCO1ACCU:NCO1ACCH:NCO1ACCL. The 24 bits are reserved but not all are used. This register updates in real time, asynchronously to the CPU; there is no provision to guarantee atomic access to this 24-bit space using an 8-bit bus. Writing to this register while the module is operating will produce undefined results.

## REGISTER 28-6: NCO1INCL: NCO1 INCREMENT REGISTER – LOW BYTE<sup>(1,2)</sup>

|          |         |         |         |         |         |         |         |
|----------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0  | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-1/1 |
| INC<7:0> |         |         |         |         |         |         |         |
| bit 7    |         |         |         | bit 0   |         |         |         |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0 **INC<7:0>:** NCO1 Increment, Low Byte

**Note 1:** The logical increment spans NCO1INCUC:NCO1INCH:NCO1INCL.

**2:** NCO1INC is double-buffered as INCBUF; INCBUF is updated on the next falling edge of NCOCLK after writing to NCO1INCL; NCO1INCUC and NCO1INCH should be written prior to writing NCO1INCL.

## REGISTER 28-7: NCO1INCH: NCO1 INCREMENT REGISTER – HIGH BYTE<sup>(1)</sup>

|           |         |         |         |         |         |         |         |
|-----------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0   | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| INC<15:8> |         |         |         |         |         |         |         |
| bit 7     |         |         |         | bit 0   |         |         |         |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0 **INC<15:8>:** NCO1 Increment, High Byte

**Note 1:** The logical increment spans NCO1INCUC:NCO1INCH:NCO1INCL.

# PIC18(L)F25/26K83

**REGISTER 28-8: NCO1INCUI: NCO1 INCREMENT REGISTER – UPPER BYTE<sup>(1)</sup>**

|           |     |     |     |         |         |         |         |
|-----------|-----|-----|-----|---------|---------|---------|---------|
| U-0       | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| INC<19:6> |     |     |     |         |         |         |         |
| bit 7     |     |     |     | bit 0   |         |         |         |

**Legend:**

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-4      **Unimplemented:** Read as '0'

bit 3-0      **INC<19:16>:** NCO1 Increment, Upper Byte

**Note 1:** The logical increment spans NCO1INCUI:NCO1INCH:NCO1INCL.

**TABLE 28-1: SUMMARY OF REGISTERS ASSOCIATED WITH NCO**

| Name      | Bit 7         | Bit 6 | Bit 5 | Bit 4 | Bit 3          | Bit 2      | Bit 1 | Bit 0 | Register on Page |
|-----------|---------------|-------|-------|-------|----------------|------------|-------|-------|------------------|
| NCO1CON   | N1EN          | —     | N1OUT | N1POL | —              | —          | —     | N1PFM | 440              |
| NCO1CLK   | N1PWS<2:0>    |       |       | —     | —              | N1CKS<2:0> |       |       | 441              |
| NCO1ACCL  | NCO1ACC<7:0>  |       |       |       |                |            |       |       | 442              |
| NCO1ACCH  | NCO1ACC<15:8> |       |       |       |                |            |       |       | 442              |
| NCO1ACCU  | —             | —     | —     | —     | NCO1ACC<19:16> |            |       |       | 443              |
| NCO1INCL  | NCO1INC<7:0>  |       |       |       |                |            |       |       | 443              |
| NCO1INCH  | NCO1INC<15:8> |       |       |       |                |            |       |       | 443              |
| NCO1INCUI | —             | —     | —     | —     | NCO1INC<19:16> |            |       |       | 444              |

**Legend:** — = unimplemented read as '0'. Shaded cells are not used for NCO module.

## 29.0 ZERO-CROSS DETECTION (ZCD) MODULE

The ZCD module detects when an A/C signal crosses through the ground potential. The actual zero-crossing threshold is the zero-crossing reference voltage, VCPINV, which is typically 0.75V above ground.

The connection to the signal to be detected is through a series current-limiting resistor. The module applies a current source or sink to the ZCD pin to maintain a constant voltage on the pin, thereby preventing the pin voltage from forward biasing the ESD protection diodes. When the applied voltage is greater than the reference voltage, the module sinks current. When the applied voltage is less than the reference voltage, the module sources current. The current source and sink action keeps the pin voltage constant over the full range of the applied voltage. The ZCD module is shown in the simplified block diagram [Figure 29-2](#).

The ZCD module is useful when monitoring an A/C waveform for, but not limited to, the following purposes:

- A/C period measurement
- Accurate long term time measurement
- Dimmer phase delayed drive
- Low EMI cycle switching

## 29.1 External Resistor Selection

The ZCD module requires a current-limiting resistor in series with the external voltage source. The impedance and rating of this resistor depends on the external source peak voltage. Select a resistor value that will drop all of the peak voltage when the current through the resistor is nominally 300  $\mu$ A. Refer to [Equation 29-1](#) and [Figure 29-1](#). Make sure that the ZCD I/O pin internal weak pull-up is disabled so it does not interfere with the current source and sink.

### EQUATION 29-1: EXTERNAL RESISTOR

$$R_{\text{SERIES}} = \frac{V_{\text{PEAK}}}{3 \times 10^{-4}}$$

FIGURE 29-1: EXTERNAL VOLTAGE

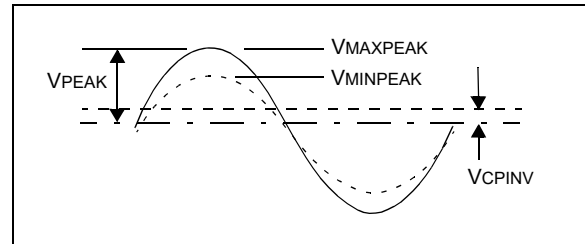
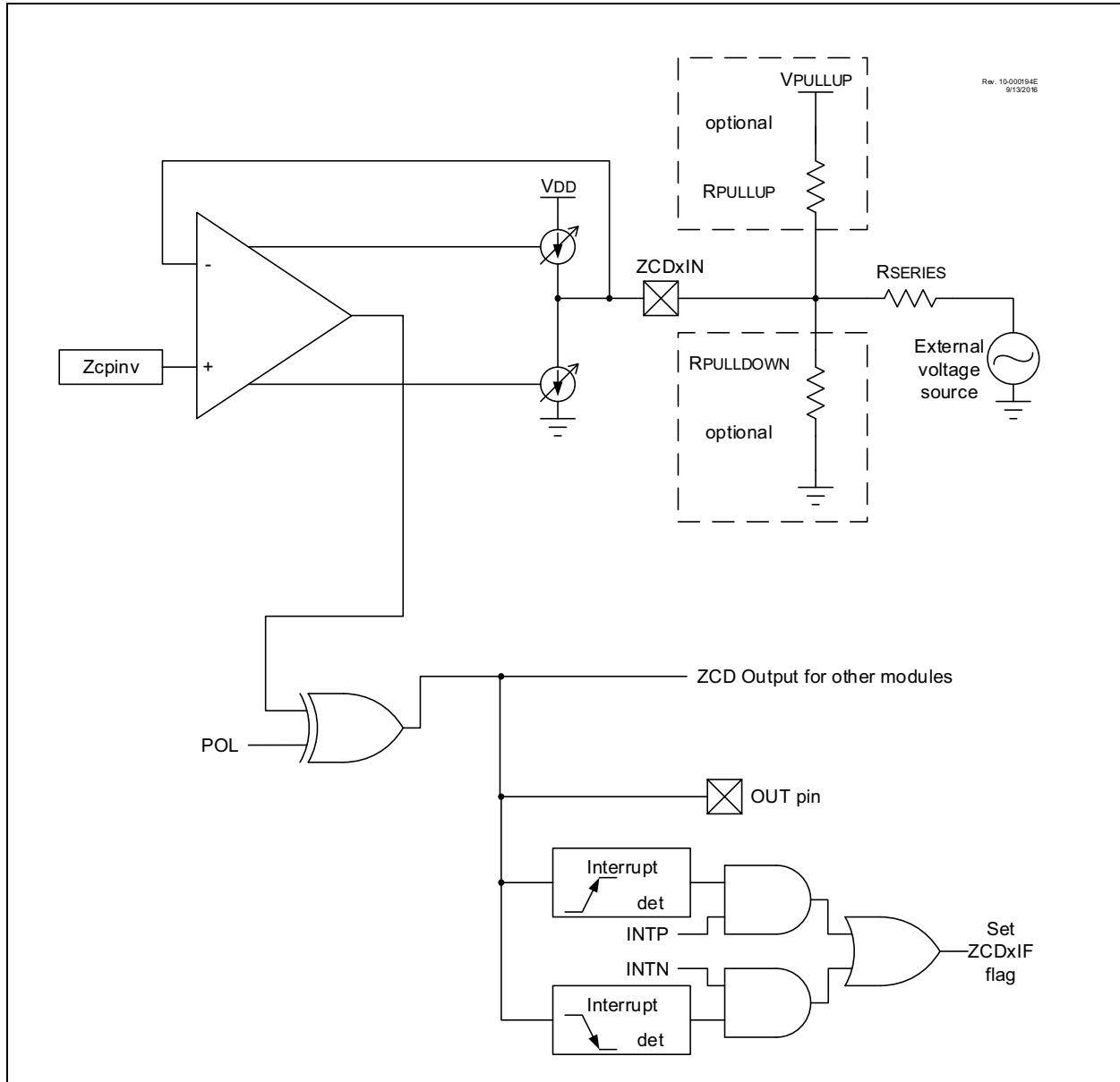


**FIGURE 29-2: SIMPLIFIED ZCD BLOCK DIAGRAM**



## 29.2 ZCD Logic Output

The ZCD module includes a Status bit, which can be read to determine whether the current source or sink is active. The OUT bit of the ZCDCON register is set when the current sink is active, and cleared when the current source is active. The OUT bit is affected by the polarity bit, even if the module is disabled.

The OUT signal can also be used as input to other modules. This is controlled by the registers of the corresponding module. OUT can be used as follows:

- Gate source for TMR1/3/5
- Clock source for TMR2/4/6
- Reset source for TMR2/4/6

## 29.3 ZCD Logic Polarity

The POL bit of the ZCDCON register inverts the OUT bit relative to the current source and sink output. When the POL bit is set, a OUT high indicates that the current source is active, and a low output indicates that the current sink is active.

The POL bit affects the ZCD interrupts.

## 29.4 ZCD Interrupts

An interrupt will be generated upon a change in the ZCD logic output when the appropriate interrupt enables are set. A rising edge detector and a falling edge detector are present in the ZCD for this purpose.

The ZCDIF bit of the respective PIR register will be set when either edge detector is triggered and its associated enable bit is set. The INTP enables rising edge interrupts and the INTN bit enables falling edge interrupts. Both are located in the ZCDCON register. Priority of the interrupt can be changed if the IPEN bit of the INTCON register is set. The ZCD interrupt can be made high or low priority by setting or clearing the ZCDIP bit of the respective IPR register.

To fully enable the interrupt, the following bits must be set:

- ZCDIE bit of the respective PIE register
- INTP bit of the ZCDCON register (for a rising edge detection)
- INTN bit of the ZCDCON register (for a falling edge detection)
- GIE bits of the INTCON0 register

Changing the POL bit can cause an interrupt, regardless of the level of the SEN bit.

The ZCDIF bit of the respective PIR register must be cleared in software as part of the interrupt service. If another edge is detected while this flag is being cleared, the flag will still be set at the end of the sequence.

## 29.5 Correcting for VCPINV offset

The actual voltage at which the ZCD switches is the reference voltage at the noninverting input of the ZCD op amp. For external voltage source waveforms other than square waves, this voltage offset from zero causes the zero-cross event to occur either too early or too late. When the waveform is varying relative to VSS, then the zero cross is detected too early as the waveform falls and too late as the waveform rises. When the waveform is varying relative to VDD, then the zero cross is detected too late as the waveform rises and too early as the waveform falls. The actual offset time can be determined for sinusoidal waveforms with the corresponding equations shown in [Equation 29-2](#).

### EQUATION 29-2: ZCD EVENT OFFSET

When External Voltage Source is relative to VSS:

$$T_{OFFSET} = \frac{\arcsin\left(\frac{V_{CPINV}}{V_{PEAK}}\right)}{2\pi \cdot Freq}$$

When External Voltage Source is relative to VDD:

$$T_{OFFSET} = \frac{\arcsin\left(\frac{V_{DD} - V_{CPINV}}{V_{PEAK}}\right)}{2\pi \cdot Freq}$$

This offset time can be compensated for by adding a pull-up or pull-down biasing resistor to the ZCD pin. A pull-up resistor is used when the external voltage source is varying relative to VSS. A pull-down resistor is used when the voltage is varying relative to VDD. The resistor adds a bias to the ZCD pin so that the target external voltage source must go to zero to pull the pin voltage to the VCPINV switching voltage. The pull-up or pull-down value can be determined with the equations shown in [Equation 29-3](#) or [Equation 29-4](#).

### EQUATION 29-3: ZCD PULL-UP/DOWN

When External Signal is relative to VSS:

$$R_{PULLUP} = \frac{R_{SERIES}(V_{PULLUP} - V_{CPINV})}{V_{CPINV}}$$

When External Signal is relative to VDD:

$$R_{PULLDOWN} = \frac{R_{SERIES}(V_{CPINV})}{(V_{DD} - V_{CPINV})}$$

Measuring VCPINV can be difficult, especially when the waveform is relative to VDD. However, by combining Equations 29-2 and 29-3, the resistor value can be determined from the time difference between the ZCD\_output high and low intervals. Note that the time difference,  $\Delta T$ , is  $4 \cdot T_{\text{OFFSET}}$ . The equation for determining the pull-up and pull-down resistor values from the high and low ZCD\_output periods is shown in Equation 29-4.

## EQUATION 29-4: PULL-UP/DOWN RESISTOR VALUES

$$R = R_{\text{SERIES}} \left( \frac{V_{\text{BIAS}}}{V_{\text{PEAK}} \left( \sin \left( \pi F_{\text{REQ}} \frac{(\Delta T)}{2} \right) \right)} - 1 \right)$$

R is pull-up or pull-down resistor.

VBIAS is VPULLUP when R is pull-up or VDD when R is pull-down.

$\Delta T$  is the ZCDOUT high and low period difference.

## 29.6 Handling VPEAK Variations

If the peak amplitude of the external voltage is expected to vary, the series resistor must be selected to keep the ZCD current source and sink below the design maximum range of  $\pm 600 \mu\text{A}$  and above a reasonable minimum range. A general rule of thumb is that the maximum peak voltage can be no more than six times the minimum peak voltage. To ensure that the maximum current does not exceed  $\pm 600 \mu\text{A}$  and the minimum is at least  $\pm 100 \mu\text{A}$ , compute the series resistance as shown in Equation 29-5. The compensating pull-up for this series resistance can be determined with Equation 29-3 because the pull-up value is not dependent to the peak voltage.

## EQUATION 29-5: SERIES R FOR V RANGE

$$R_{\text{SERIES}} = \frac{V_{\text{MAXPEAK}} + V_{\text{MINPEAK}}}{7 \times 10^{-4}}$$

## 29.7 Operation During Sleep

The ZCD current sources and interrupts are unaffected by Sleep.

## 29.8 Effects of a Reset

The ZCD circuit can be configured to default to the active or inactive state on Power-on-Reset (POR). When the  $\overline{\text{ZCD}}$  Configuration bit is cleared, the ZCD circuit will be active at POR. When the  $\overline{\text{ZCD}}$  Configuration bit is set, the SEN bit of the ZCDCON register must be set to enable the ZCD module.

## 29.9 Disabling the ZCD Module

The ZCD module can be disabled in two ways:

1. Configuration Word 2H has the  $\overline{\text{ZCD}}$  bit which disables the ZCD module when set, but it can be enabled using the SEN bit of the ZCDCON register (Register 29-1). If the  $\overline{\text{ZCD}}$  bit is clear, the ZCD is always enabled.
2. The ZCD can also be disabled using the ZCDMD bit of the respective PMD2 register (Register 19-3). This is subject to the status of the  $\overline{\text{ZCD}}$  bit.



## 29.10 Register Definitions: ZCD Control

**REGISTER 29-1: ZCDCON: ZERO-CROSS DETECT CONTROL REGISTER**

|         |     |     |         |     |     |         |         |
|---------|-----|-----|---------|-----|-----|---------|---------|
| R/W-0/0 | U-0 | R-x | R/W-0/0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 |
| SEN     | —   | OUT | POL     | —   | —   | INTP    | INTN    |
| bit 7   |     |     |         |     |     |         | bit 0   |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

|         |                                                                                                                                                                                                                                                |
|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| bit 7   | <b>SEN:</b> Zero-Cross Detect Software Enable bit<br>This bit is ignored when ZCDSEN Configuration bit is set.<br>1 = Zero-cross detect is enabled.<br>0 = Zero-cross detect is disabled. ZCD pin operates according to PPS and TRIS controls. |
| bit 6   | <b>Unimplemented:</b> Read as '0'                                                                                                                                                                                                              |
| bit 5   | <b>OUT:</b> Zero-Cross Detect Data Output bit<br><u>ZCDPOL bit = 0:</u><br>1 = ZCD pin is sinking current<br>0 = ZCD pin is sourcing current<br><u>ZCDPOL bit = 1:</u><br>1 = ZCD pin is sourcing current<br>0 = ZCD pin is sinking current    |
| bit 4   | <b>POL:</b> Zero-Cross Detect Polarity bit<br>1 = ZCD logic output is inverted<br>0 = ZCD logic output is not inverted                                                                                                                         |
| bit 3-2 | <b>Unimplemented:</b> Read as '0'                                                                                                                                                                                                              |
| bit 1   | <b>INTP:</b> Zero-Cross Detect Positive-Going Edge Interrupt Enable bit<br>1 = ZCDIF bit is set on low-to-high ZCD_output transition<br>0 = ZCDIF bit is unaffected by low-to-high ZCD_output transition                                       |
| bit 0   | <b>INTN:</b> Zero-Cross Detect Negative-Going Edge Interrupt Enable bit<br>1 = ZCDIF bit is set on high-to-low ZCD_output transition<br>0 = ZCDIF bit is unaffected by high-to-low ZCD_output transition                                       |

**TABLE 29-1: SUMMARY OF REGISTERS ASSOCIATED WITH THE ZCD MODULE**

| Name   | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|--------|-------|-------|-------|-------|-------|-------|-------|-------|------------------|
| ZCDCON | SEN   | —     | OUT   | POL   | —     | —     | INTP  | INTN  | 449              |

**Legend:** — = unimplemented, read as '0'. Shaded cells are unused by the ZCD module.

## 30.0 DATA SIGNAL MODULATOR (DSM) MODULE

The Data Signal Modulator (DSM) is a peripheral which allows the user to mix a data stream, also known as a modulator signal, with a carrier signal to produce a modulated output.

Both the carrier and the modulator signals are supplied to the DSM module either internally, from the output of a peripheral, or externally through an input pin.

The modulated output signal is generated by performing a logical “AND” operation of both the carrier and modulator signals and then provided to the MDOUT pin.

The carrier signal is comprised of two distinct and separate signals. A carrier high (CARH) signal and a carrier low (CARL) signal. During the time in which the modulator (MOD) signal is in a logic high state, the DSM mixes the carrier high signal with the modulator signal. When the modulator signal is in a logic low state, the DSM mixes the carrier low signal with the modulator signal.

Using this method, the DSM can generate the following types of Key Modulation schemes:

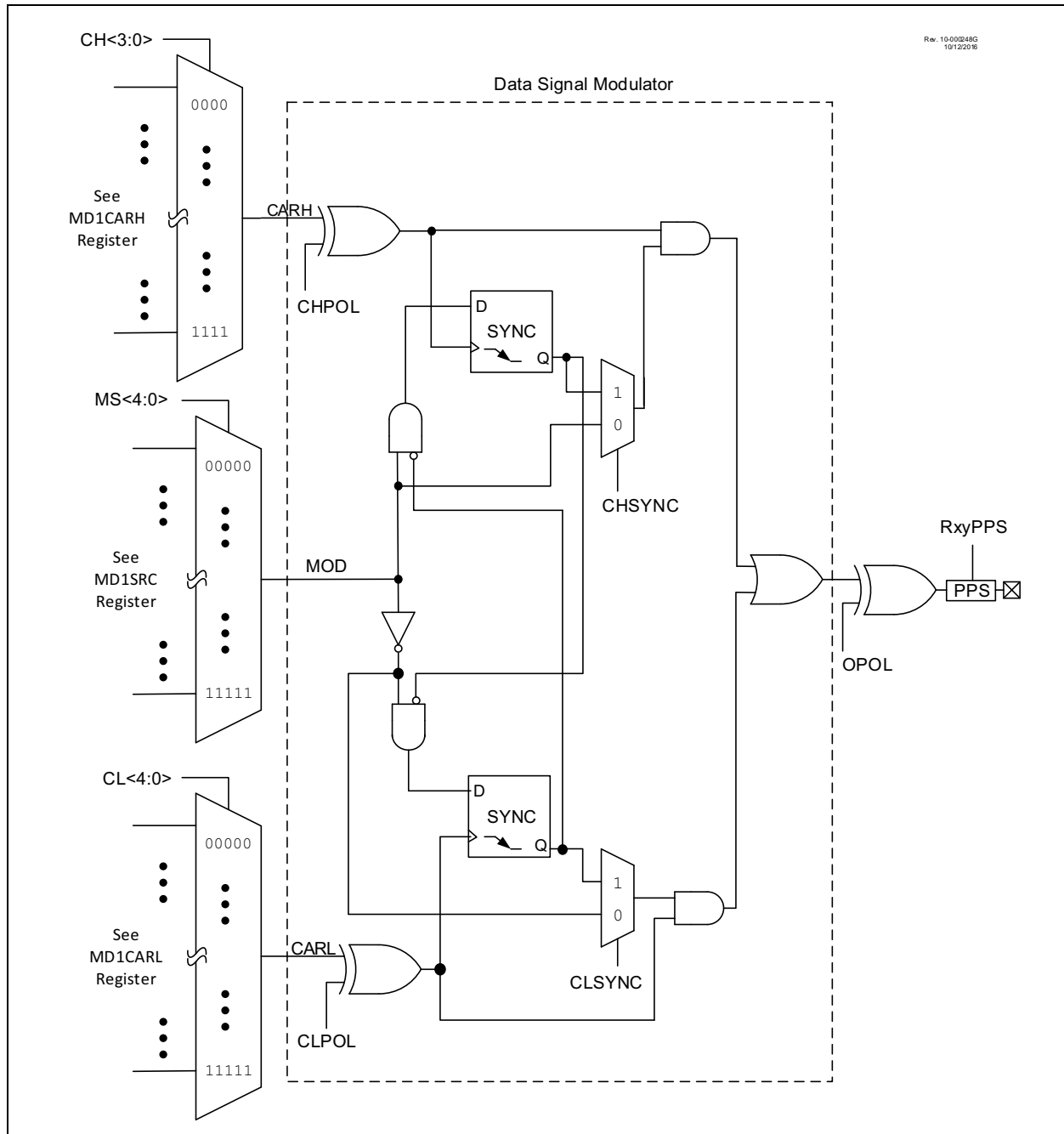
- Frequency-Shift Keying (FSK)
- Phase-Shift Keying (PSK)
- On-Off Keying (OOK)

Additionally, the following features are provided within the DSM module:

- Carrier Synchronization
- Carrier Source Polarity Select
- Programmable Modulator Data
- Modulated Output Polarity Select
- Peripheral Module Disable, which provides the ability to place the DSM module in the lowest power consumption mode

[Figure 30-1](#) shows a Simplified Block Diagram of the Data Signal Modulator peripheral.

**FIGURE 30-1: SIMPLIFIED BLOCK DIAGRAM OF THE DATA SIGNAL MODULATOR**



## 30.1 DSM Operation

The DSM module can be enabled by setting the EN bit in the MD1CON0 register. Clearing the EN bit in the MD1CON0 register, disables the DSM module output and switches the carrier high and carrier low signals to the default option of MD1CARHPPS and MD1CARLPPS, respectively. The modulator signal source is also switched to the BIT in the MD1CON0 register.

The values used to select the carrier high, carrier low, and modulator sources held by the Modulation Source, Modulation High Carrier, and Modulation Low Carrier control registers are not affected when the EN bit is cleared and the DSM module is disabled. The values inside these registers remain unchanged while the DSM is inactive. The sources for the carrier high, carrier low and modulator signals will once again be selected when the EN bit is set and the DSM module is again enabled and active.

## 30.2 Modulator Signal Sources

The modulator signal can be supplied from the sources specified in [Table 30-3](#).

The modulator signal is selected by configuring the MS<4:0> bits in the MD1SRC register.

## 30.3 Carrier Signal Sources

The carrier high signal and carrier low signal can be supplied from the sources specified in [Table 30-1](#).

The carrier high signal is selected by configuring the CH<4:0> bits in the MD1CARH register. The carrier low signal is selected by configuring the CL<4:0> bits in the MD1CARL register.

## 30.4 Carrier Synchronization

During the time when the DSM switches between carrier high and carrier low signal sources, the carrier data in the modulated output signal can become truncated. To prevent this, the carrier signal can be synchronized to the modulator signal. When synchronization is enabled, the carrier pulse that is being mixed at the time of the transition is allowed to transition low before the DSM switches over to the next carrier source.

Synchronization is enabled separately for the carrier high and carrier low signal sources. Synchronization for the carrier high signal is enabled by setting the CHSYNC bit in the MD1CON1 register. Synchronization for the carrier low signal is enabled by setting the CLSYNC bit in the MD1CON1 register.

[Figure 30-2](#) through [Figure 30-6](#) show timing diagrams of using various synchronization methods.

**FIGURE 30-2: On Off Keying (OOK) Synchronization**

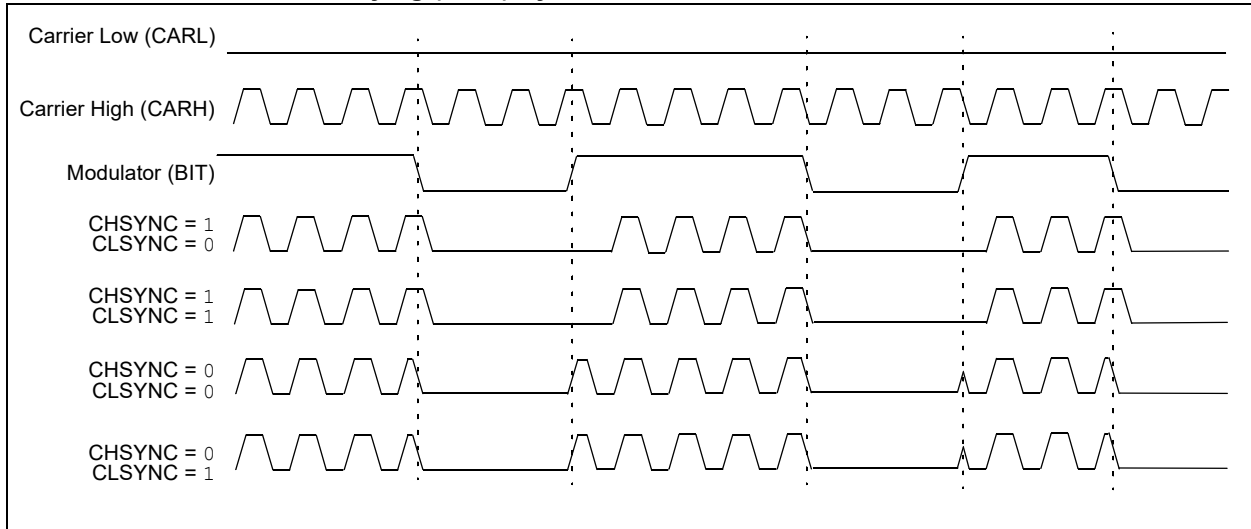


**FIGURE 30-3: No Synchronization (CHSYNC = 0, CLSYNC = 0)**

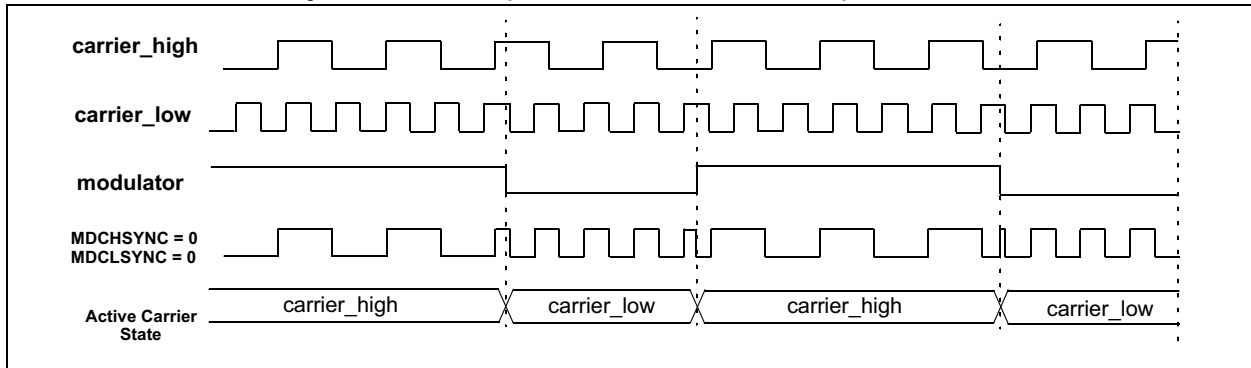


**FIGURE 30-4: Carrier High Synchronization (CHSYNC = 1, CLSYNC = 0)**

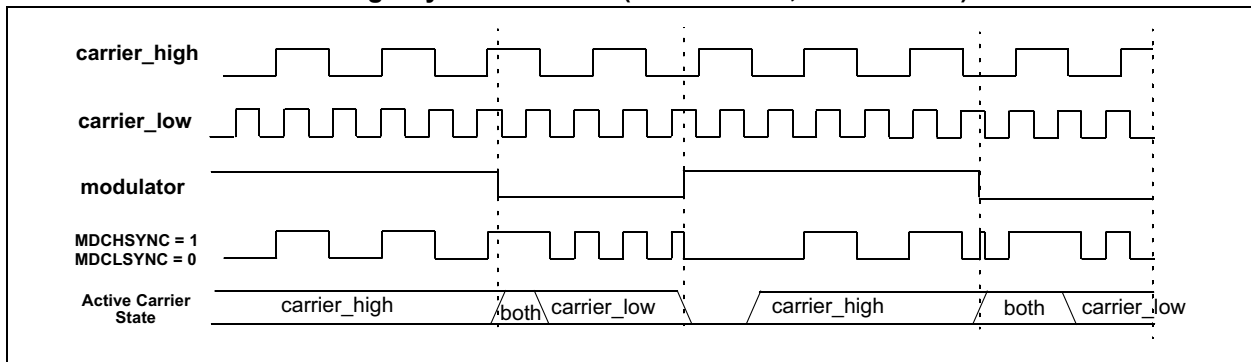


**FIGURE 30-5: Carrier Low Synchronization (CHSYNC = 0, CLSYNC = 1)**

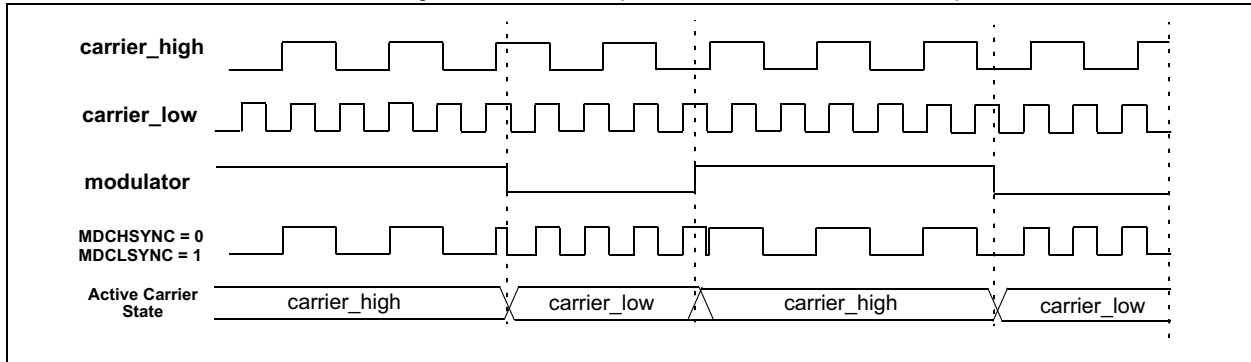
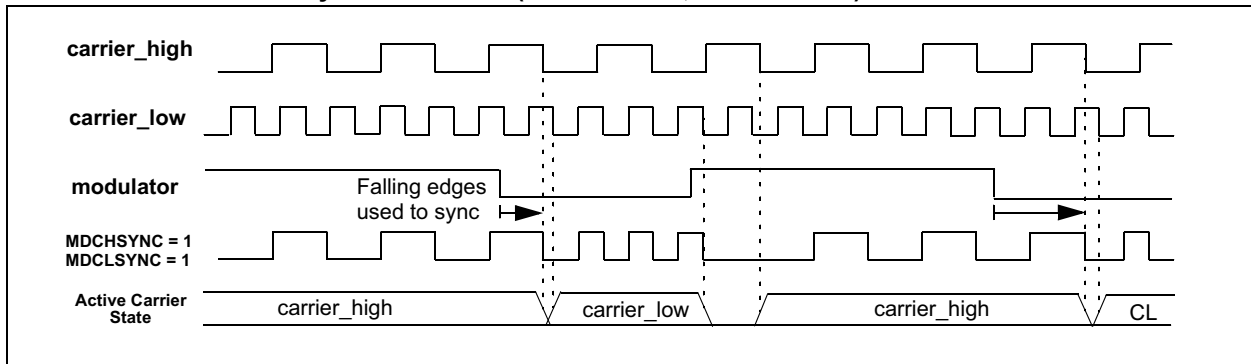


**FIGURE 30-6: Full Synchronization (CHSYNC = 1, CLSYNC = 1)**



## 30.5 Carrier Source Polarity Select

The signal provided from any selected input source for the carrier high and carrier low signals can be inverted. Inverting the signal for the carrier high source is enabled by setting the CHPOL bit of the MD1CON1 register. Inverting the signal for the carrier low source is enabled by setting the CLPOL bit of the MD1CON1 register.

## 30.6 Programmable Modulator Data

The BIT of the MD1CON0 register can be selected as the source for the modulator signal. This gives the user the ability to program the value used for modulation.

## 30.7 Modulated Output Polarity

The modulated output signal provided on the DSM pin can also be inverted. Inverting the modulated output signal is enabled by setting the OPOL bit of the MD1CON0 register.

## 30.8 Operation in Sleep Mode

The DSM module is not affected by Sleep mode. The DSM can still operate during Sleep, if the Carrier and Modulator input sources are also still operable during Sleep. Refer to [Section 10.0 “Power-Saving Operation Modes”](#) for more details.

## 30.9 Effects of a Reset

Upon any device Reset, the DSM module is disabled. The user's firmware is responsible for initializing the module before enabling the output. The registers are reset to their default values.

## 30.10 Peripheral Module Disable

The DSM module can be completely disabled using the PMD module to achieve maximum power saving. The DSMMD bit of PMD6 ([Register 19-7](#)) when set disables the DSM module completely. When enabled again all the registers of the DSM module default to POR status.

## 30.11 Register Definitions: Modulation Control

Long bit name prefixes for the Modulation peripheral is shown below. Refer to [Section 1.3.2.2 “Long Bit Names”](#) for more information.

| Peripheral | Bit Name Prefix |
|------------|-----------------|
| MD1        | MD1             |

### REGISTER 30-1: MD1CON0: MODULATION CONTROL REGISTER 0

|         |     |       |         |     |     |     |         |
|---------|-----|-------|---------|-----|-----|-----|---------|
| R/W-0/0 | U-0 | R-0/0 | R/W-0/0 | U-0 | U-0 | U-0 | R/W-0/0 |
| EN      | —   | OUT   | OPOL    | —   | —   | —   | BIT     |
| bit 7   |     |       |         |     |     |     | bit 0   |

#### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

- bit 7 **EN:** Modulator Module Enable bit  
 1 = Modulator module is enabled and mixing input signals  
 0 = Modulator module is disabled and has no output
- bit 6 **Unimplemented:** Read as '0'
- bit 5 **OUT:** Modulator Output bit  
 Displays the current output value of the Modulator module.<sup>(1)</sup>
- bit 4 **OPOL:** Modulator Output Polarity Select bit  
 1 = Modulator output signal is inverted; idle high output  
 0 = Modulator output signal is not inverted; idle low output
- bit 3-1 **Unimplemented:** Read as '0'
- bit 0 **BIT:** Allows software to manually set modulation source input to module<sup>(2)</sup>  
 1 = Modulator selects Carrier High  
 0 = Modulator selects Carrier Low

**Note 1:** The modulated output frequency can be greater and asynchronous from the clock that updates this register bit, the bit value may not be valid for higher speed modulator or carrier signals.

**2:** BIT bit must be selected as the modulation source in the MD1SRC register for this operation.



## REGISTER 30-2: MD1CON1: MODULATION CONTROL REGISTER 1

|       |     |         |         |     |     |         |         |
|-------|-----|---------|---------|-----|-----|---------|---------|
| U-0   | U-0 | R/W-0/0 | R/W-0/0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 |
| —     | —   | CHPOL   | CHSYNC  | —   | —   | CLPOL   | CLSYNC  |
| bit 7 |     |         |         |     |     | bit 0   |         |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-6 **Unimplemented:** Read as '0'

bit 5 **CHPOL:** Modulator High Carrier Polarity Select bit

1 = Selected high carrier signal is inverted

0 = Selected high carrier signal is not inverted

bit 4 **CHSYNC:** Modulator High Carrier Synchronization Enable bit

1 = Modulator waits for a falling edge on the high time carrier signal before allowing a switch to the low time carrier

0 = Modulator output is not synchronized to the high time carrier signal<sup>(1)</sup>

bit 3-2 **Unimplemented:** Read as '0'

bit 1 **CLPOL:** Modulator Low Carrier Polarity Select bit

1 = Selected low carrier signal is inverted

0 = Selected low carrier signal is not inverted

bit 0 **CLSYNC:** Modulator Low Carrier Synchronization Enable bit

1 = Modulator waits for a falling edge on the low time carrier signal before allowing a switch to the high time carrier

0 = Modulator output is not synchronized to the low time carrier signal<sup>(1)</sup>

**Note 1:** Narrowed carrier pulse widths or spurs may occur in the signal stream if the carrier is not synchronized.

## REGISTER 30-3: MD1CARH: MODULATION HIGH CARRIER CONTROL REGISTER

| U-0   | U-0 | U-0 | R/W-0/0                | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|-------|-----|-----|------------------------|---------|---------|---------|---------|
| —     | —   | —   | CH<4:0> <sup>(1)</sup> |         |         |         |         |
| bit 7 |     |     |                        |         |         |         |         |
|       |     |     |                        |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-5 **Unimplemented:** Read as '0'

bit 4-0 **CH<4:0>:** Modulator Carrier High Selection bits<sup>(1)</sup>

See [Table 30-1](#) for signal list

**Note 1:** Unused selections provide an input value.

## REGISTER 30-4: MD1CARL: MODULATION LOW CARRIER CONTROL REGISTER

| U-0   | U-0 | U-0 | R/W-0/0                | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|-------|-----|-----|------------------------|---------|---------|---------|---------|
| —     | —   | —   | CL<4:0> <sup>(1)</sup> |         |         |         |         |
| bit 7 |     |     |                        |         |         |         |         |
|       |     |     |                        |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-5 **Unimplemented:** Read as '0'

bit 4-0 **CL<4:0>:** Modulator Carrier Low Input Selection bits<sup>(1)</sup>

See [Table 30-1](#) for signal list

**Note 1:** Unused selections provide a zero as the input value.

**TABLE 30-1: MD1CARH/MD1CARL SELECTION MUX CONNECTIONS**

| MD1CARH     |       |                            | MD1CARL     |       |                            |
|-------------|-------|----------------------------|-------------|-------|----------------------------|
| CH<4:0>     |       | Connection                 | CL<4:0>     |       | Connection                 |
| 11111-10011 | 31-19 | Reserved                   | 11111-10011 | 31-19 | Reserved                   |
| 10010       | 18    | CLC4OUT                    | 10010       | 18    | CLC4OUT                    |
| 10001       | 17    | CLC3OUT                    | 10001       | 17    | CLC3OUT                    |
| 10000       | 16    | CLC2OUT                    | 10000       | 16    | CLC2OUT                    |
| 01111       | 15    | CLC1OUT                    | 01111       | 15    | CLC1OUT                    |
| 01110       | 14    | NCO1OUT                    | 01110       | 14    | NCO1OUT                    |
| 01101-01100 | 13-12 | Reserved                   | 01101-01100 | 13-12 | Reserved                   |
| 01011       | 11    | PWM8 OUT                   | 01011       | 11    | PWM8 OUT                   |
| 01010       | 10    | PWM7 OUT                   | 01010       | 10    | PWM7 OUT                   |
| 01001       | 9     | PWM6 OUT                   | 01001       | 9     | PWM6 OUT                   |
| 01000       | 8     | PWM5 OUT                   | 01000       | 8     | PWM5 OUT                   |
| 00111       | 7     | CCP4 OUT                   | 00111       | 7     | CCP4 OUT                   |
| 00110       | 6     | CCP3 OUT                   | 00110       | 6     | CCP3 OUT                   |
| 00101       | 5     | CCP2 OUT                   | 00101       | 5     | CCP2 OUT                   |
| 00100       | 4     | CCP1 OUT                   | 00100       | 4     | CCP1 OUT                   |
| 00011       | 3     | CLKREF output              | 00011       | 3     | CLKREF output              |
| 00010       | 2     | HFINTOSC                   | 00010       | 2     | HFINTOSC                   |
| 00001       | 1     | FOSC (system clock)        | 00001       | 1     | FOSC (system clock)        |
| 00000       | 0     | Pin selected by MD1CARHPPS | 00000       | 0     | Pin selected by MD1CARLPPS |

**REGISTER 30-5: MD1SRC: MODULATION SOURCE CONTROL REGISTER**

|       |     |     |         |         |         |         |         |
|-------|-----|-----|---------|---------|---------|---------|---------|
| U-0   | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| —     | —   | —   | MS<4:0> |         |         |         |         |
| bit 7 |     |     | bit 0   |         |         |         |         |

**Legend:**

R = Readable bit                      W = Writable bit                      U = Unimplemented bit, read as '0'  
 u = Bit is unchanged                  x = Bit is unknown                      -n/n = Value at POR and BOR/Value at all other Resets  
 '1' = Bit is set                          '0' = Bit is cleared

bit 7-5                      **Unimplemented:** Read as '0'  
 bit 4-0                      **MS<4:0>:** Modulator Source Selection bits<sup>(1)</sup>  
                                     See [Table 30-2](#) for signal list

**Note 1:** Unused selections provide a zero as the input value.

**TABLE 30-2: MD1SRC SELECTION MUX CONNECTIONS**

| MS<4:0> |    | Connection               |
|---------|----|--------------------------|
| 1 1111  | 31 | Reserved                 |
| 1 1000  | 24 |                          |
| 1 0111  | 23 | CAN_tx0                  |
| 1 0110  | 22 | SPI1 SDO                 |
| 1 0101  | 21 | Reserved                 |
| 1 0100  | 20 | UART2 TX                 |
| 1 0011  | 19 | UART1 TX                 |
| 1 0010  | 18 | CLC4 OUT                 |
| 1 0001  | 17 | CLC3 OUT                 |
| 1 0000  | 16 | CLC2 OUT                 |
| 0 1111  | 15 | CLC1 OUT                 |
| 0 1110  | 14 | CMP2 OUT                 |
| 0 1101  | 13 | CMP1 OUT                 |
| 0 1100  | 12 | NCO1 OUT                 |
| 0 1011  | 11 | Reserved                 |
| 0 1010  | 10 | Reserved                 |
| 0 1001  | 9  | PWM8 OUT                 |
| 0 1000  | 8  | PWM7 OUT                 |
| 0 0111  | 7  | PWM6 OUT                 |
| 0 0110  | 6  | PWM5 OUT                 |
| 0 0101  | 5  | CCP4 OUT                 |
| 0 0100  | 4  | CCP3 OUT                 |
| 0 0011  | 3  | CCP2 OUT                 |
| 0 0010  | 2  | CCP1 OUT                 |
| 0 0001  | 1  | DSM1 BIT                 |
| 0 0000  | 0  | Pin selected by MDSRCPPS |

**TABLE 30-3: SUMMARY OF REGISTERS ASSOCIATED WITH DATA SIGNAL MODULATOR MODE**

| Name    | Bit 7 | Bit 6 | Bit 5 | Bit 4  | Bit 3     | Bit 2    | Bit 1 | Bit 0  | Register on Page    |
|---------|-------|-------|-------|--------|-----------|----------|-------|--------|---------------------|
| MD1CON0 | EN    | —     | OUT   | OPOL   | —         | —        | —     | BIT    | <a href="#">456</a> |
| MD1CON1 | —     | —     | CHPOL | CHSYNC | —         | —        | CLPOL | CLSYNC | <a href="#">457</a> |
| MD1CARH | —     | —     | —     | —      | —         | CHS<2:0> |       |        | <a href="#">458</a> |
| MD1CARL | —     | —     | —     | —      | —         | CLS<2:0> |       |        | <a href="#">458</a> |
| MDSRC   | —     | —     | —     | —      | SRCS<3:0> |          |       |        | <a href="#">459</a> |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used in the Data Signal Modulator mode.

## 31.0 UNIVERSAL ASYNCHRONOUS RECEIVER TRANSMITTER (UART) WITH PROTOCOL SUPPORT

The Universal Asynchronous Receiver Transmitter (UART) module is a serial I/O communications peripheral. It contains all the clock generators, shift registers and data buffers necessary to perform an input or output serial data transfer, independent of device program execution. The UART, also known as a Serial Communications Interface (SCI), can be configured as a full-duplex asynchronous system or one of several automated protocols. Full-Duplex mode is useful for communications with peripheral systems, such as CRT terminals and personal computers.

Supported protocols include:

- LIN Master and Slave
- DMX mode
- DALI control gear and control device

The UART module includes the following capabilities:

- Full-duplex asynchronous transmit and receive
- Two-character input buffer
- One-character output buffer
- Programmable 7-bit or 8-bit character length
- 9th bit Address detection
- 9th bit even or odd parity
- Input buffer overrun error detection
- Received character framing error detection
- Hardware and software flow control
- Automatic checksums
- Programmable 1, 1.5, and 2 Stop bits
- Programmable data polarity
- Manchester encoder/decoder
- Operation in Sleep
- Automatic detection and calibration of the baud rate
- Wake-up on Break reception
- Automatic and user timed Break period generation
- RX and TX inactivity timeouts (with Timer2)

Block diagrams of the UART transmitter and receiver are shown in [Figure 31-1](#) and [Figure 31-2](#).

The UART transmit output (TX\_out) is available to the TX pin and internally to various peripherals.

**FIGURE 31-1: UART TRANSMIT BLOCK DIAGRAM**

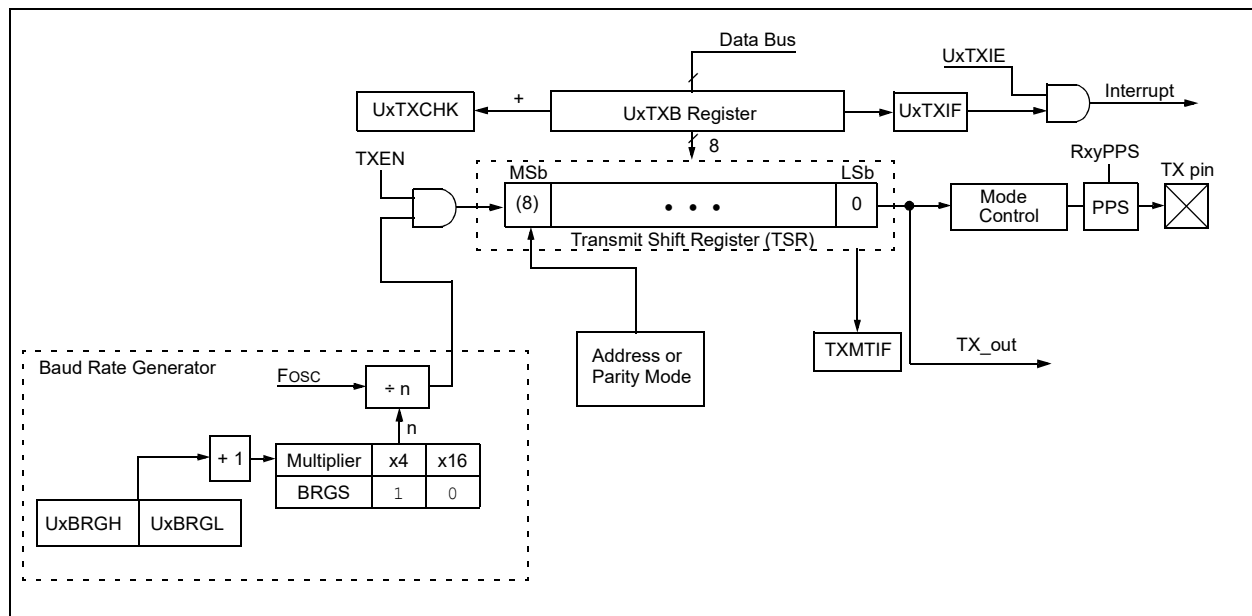
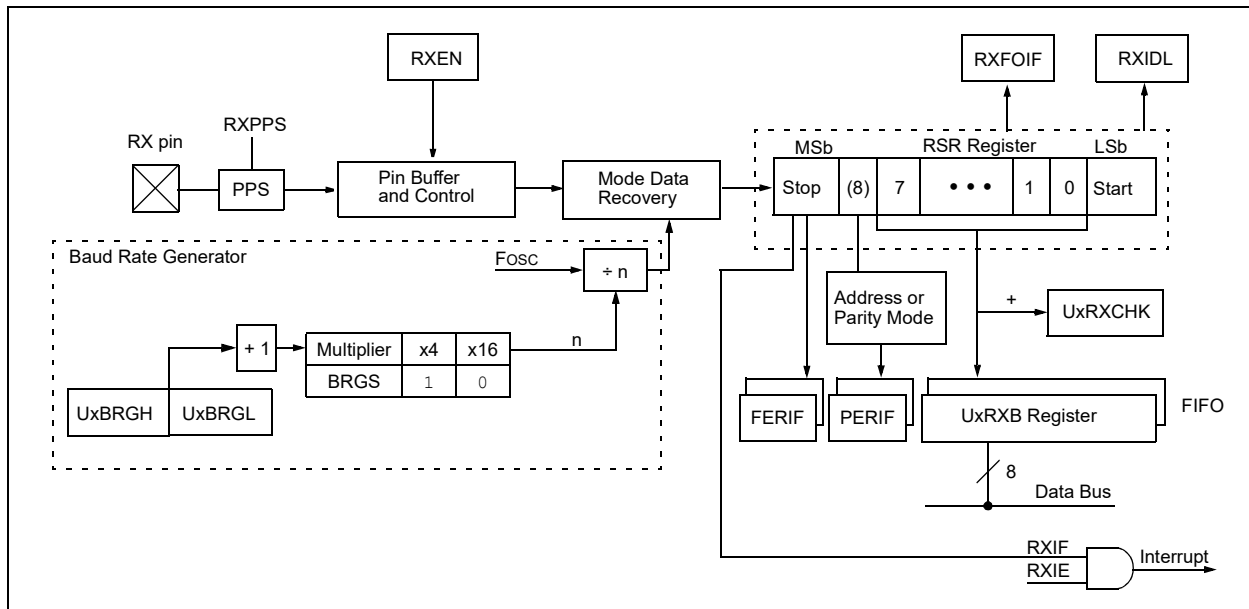


**FIGURE 31-2: UART RECEIVE BLOCK DIAGRAM**



The operation of the UART module is controlled through nineteen registers:

- Three control registers (UxCON0-UxCON2)
- Error enable and status (UxERRIE, UxERRIR, UxUIR)
- UART buffer status and control (UxFIFO)
- Three 9-bit protocol parameters (UxP1-UxP3)
- 16-bit baud rate generator (UxBRGH:L)
- Transmit buffer write (UxTXB)
- Receive buffer read (UxRXB)
- Receive checksum (UxRXCHK)
- Transmit checksum (UxTXCHK)

These registers are detailed in [Section 31.21 “Register Definitions: UART Control”](#).

## 31.1 UART I/O Pin Configuration

The RX input pin is selected with the UxRPPS register. The TX output pin is selected with each pin's RxyPPS register. When the TRIS control for the pin corresponding to the TX output is cleared, then the UART will maintain control and the logic level on the TX pin. Changing the TXPOL bit in UxCON2 will immediately change the TX pin logic level regardless of the value of EN or TXEN.

## 31.2 UART Asynchronous Modes

The UART has five asynchronous modes:

- 7-bit
- 8-bit
- 8-bit with even parity in the 9th bit
- 8-bit with odd parity in the 9th bit
- 8-bit with address indicator in the 9th bit

The UART transmits and receives data using the standard Non-Return-to-Zero (NRZ) format. NRZ is implemented with two levels: a VOH Mark state, which

represents a '1' data bit, and a VOL Space state, which represents a '0' data bit. NRZ refers to the fact that consecutively transmitted data bits of the same value stay at the output level of that bit without returning to a neutral level between each bit transmission. An NRZ transmission port idles in the Mark state. Each character transmission consists of one Start bit followed by seven or eight data bits, one optional parity or address bit, and is always terminated by one or more Stop bits. The Start bit is always a space and the Stop bits are always marks. The most common data format is eight bits with no parity. Each transmitted bit persists for a period of 1/(Baud Rate). An on-chip dedicated 16-bit Baud Rate Generator is used to derive standard baud rate frequencies from the system oscillator. See [Section 31.17 “UART Baud Rate Generator \(BRG\)”](#) for more information.

In all the Asynchronous modes, the UART transmits and receives the LSb first. The UART's transmitter and receiver are functionally independent, but share the same data format and baud rate. Parity is supported by the hardware by even and odd parity modes.

### 31.2.1 UART ASYNCHRONOUS TRANSMITTER

The UART transmitter block diagram is shown in [Figure 31-1](#). The heart of the transmitter is the serial Transmit Shift Register (TSR), which is not directly accessible by software. The TSR obtains its data from the transmit buffer, which is the UxTXB register.

## 31.2.1.1 Enabling the Transmitter

The UART transmitter is enabled for asynchronous operations by configuring the following control bits:

- TXEN = 1
- MODE<3:0> = 0h through 3h
- UxBRGH:L = desired baud rate
- UxBRGS = desired baud rate multiplier
- RxyPPS = code for desired output pin
- ON = 1

All other UART control bits are assumed to be in their default state.

Setting the TXEN bit in the UxCON0 register enables the transmitter circuitry of the UART. The MODE<3:0> bits in the UxCON0 register select the desired mode. Setting the ON bit in the UxCON1 register enables the UART. When TXEN is set and the transmitter is not idle, the TX pin is automatically configured as an output. When the transmitter is idle, the TX pin drive is relinquished to the port TRIS control. If the TX pin is shared with an analog peripheral, the analog I/O function should be disabled by clearing the corresponding ANSEL bit.

|                                                                                                                                   |
|-----------------------------------------------------------------------------------------------------------------------------------|
| <b>Note:</b> The UxTXIF Transmitter Interrupt flag is set when the TXEN enable bit is set and the UxTXB register can accept data. |
|-----------------------------------------------------------------------------------------------------------------------------------|

## 31.2.1.2 Transmitting Data

A transmission is initiated by writing a character to the UxTXB register. If this is the first character, or the previous character has been completely transmitted from the TSR, the data in the UxTXB is immediately transferred to the TSR register. If the TSR still contains all or part of a previous character, the new character data is held in the UxTXB until the previous character transmission is complete. The pending character in the UxTXB is then transferred to the TSR at the beginning of the previous character Stop bit transmission. The transmission of the Start bit, data bits and Stop bit sequence commences immediately following the completion of all of the previous character's Stop bits.

## 31.2.1.3 Transmit Data Polarity

The polarity of the transmit data is controlled with the TXPOL bit in the UxCON2 register. The default state of this bit is '0' which selects high true transmit idle and data bits. Setting the TXPOL bit to '1' will invert the transmit data, resulting in low true idle and data bits. The TXPOL bit controls transmit data polarity in all modes.

## 31.2.1.4 Transmit Interrupt Flag

The UxTXIF interrupt flag bit in the PIR register is set whenever the UART transmitter is enabled and no character is being held for transmission in the UxTXB. In other words, the UxTXIF bit is clear only when the TSR is busy with a character and a new character has been queued for transmission in the UxTXB.

The UxTXIF interrupt can be enabled by setting the UxTXIE interrupt enable bit in the PIE register. However, the UxTXIF flag bit will be set whenever the UxTXB is empty, regardless of the state of UxTXIE enable bit. The UxTXIF bit is read-only and cannot be set or cleared by software.

To use interrupts when transmitting data, set the UxTXIE bit only when there is more data to send. Clear the UxTXIE interrupt enable bit upon writing UxTXB with the last character of the transmission.

## 31.2.1.5 TSR Status

The TXMTIF bit in the UxERRIR register indicates the status of the TSR. This is a read-only bit. The TXMTIF bit is set when the TSR is empty and idle. The TXMTIF bit is cleared when a character is transferred to the TSR from the UxTXB. The TXMTIF bit remains clear until all bits, including the Stop bits, have been shifted out of the TSR and a byte is not waiting in the UxTXB register.

The TXMTIF will generate an interrupt when the TXMTIE bit in the UxERRIE register is set.

|                                                                                        |
|----------------------------------------------------------------------------------------|
| <b>Note:</b> The TSR is not mapped in data memory, so it is not available to the user. |
|----------------------------------------------------------------------------------------|

## 31.2.1.6 Transmitter 7-bit Mode

7-Bit mode is selected when the MODE<3:0> bits are set to '0001'. In 7-bit mode, only the seven Least Significant bits of the data written to UxTXB are transmitted. The Most Significant bit is ignored.

## 31.2.1.7 Transmitter Parity Modes

When the Odd or even Parity mode is selected, all data is sent as nine bits. The first eight bits are data and the 9th bit is parity. Even and odd parity is selected when the MODE<3:0> bits are set to '0011' and '0010', respectively. Parity is automatically determined by the module and inserted in the serial data stream.

## 31.2.1.8 Asynchronous Transmission Setup

1. Initialize the UxBRGH, UxBRGL register pair and the BRGS bit to achieve the desired baud rate (see [Section 31.17 “UART Baud Rate Generator \(BRG\)”](#)).
2. Set the MODE<3:0> bits to the desired Asynchronous mode.
3. Set TXPOL bit if inverted TX output is desired.
4. Enable the asynchronous serial port by setting the ON bit.
5. Enable the transmitter by setting the TXEN control bit. This will cause the UxTXIF interrupt flag to be set.
6. If the device has PPS, configure the desired I/O pin RxyPPS register with the code for TX output.
7. If interrupts are desired, set the UxTXIE interrupt enable bit in the respective PIE register. An interrupt will occur immediately provided that the GIE bits in the INTCON0 register are also set.
8. Write one byte of data into the UxTXB register. This will start the transmission.
9. Subsequent bytes may be written when the UxTXIF bit is ‘1’.

**FIGURE 31-3: ASYNCHRONOUS TRANSMISSION**

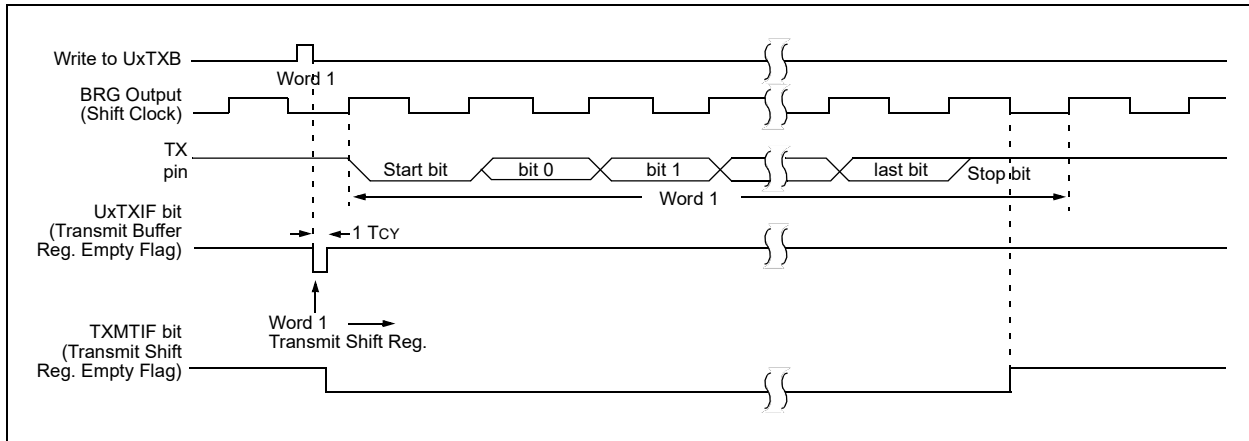
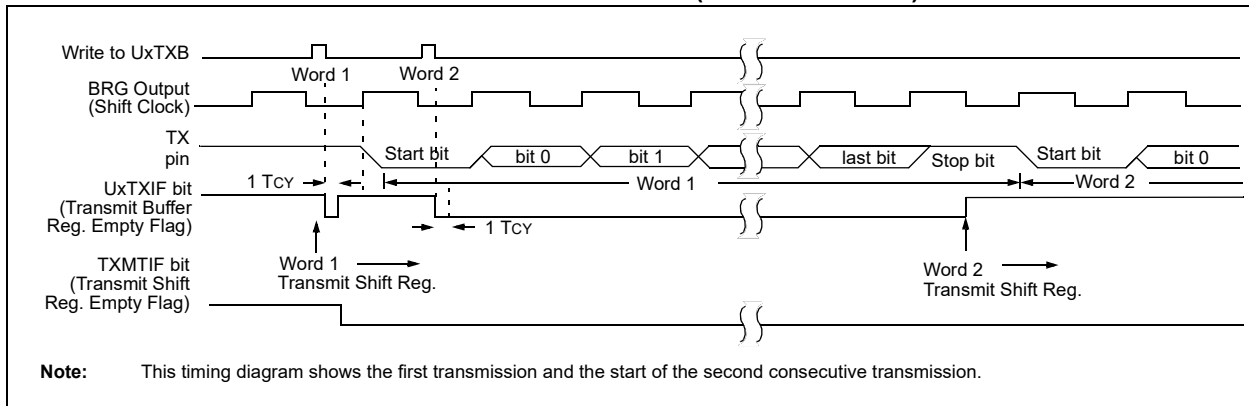


**FIGURE 31-4: ASYNCHRONOUS TRANSMISSION (BACK-TO-BACK)**





## 31.2.2 UART ASYNCHRONOUS RECEIVER

The Asynchronous mode is typically used in RS-232 systems. The receiver block diagram is shown in [Figure 31-2](#). The data is received on the RX pin and drives the data recovery block. The data recovery block is actually a high-speed shifter operating at 4 or 16 times the baud rate, whereas the serial Receive Shift Register (RSR) operates at the bit rate. When all bits of the character have been shifted in, they are immediately transferred to a two character First-In-First-Out (FIFO) memory. The FIFO buffering allows reception of two complete characters and the start of a third character before software must start servicing the UART receiver. The FIFO registers and RSR are not directly accessible by software. Access to the received data is via the UxRXB register.

### 31.2.2.1 Enabling the Receiver

The UART receiver is enabled for asynchronous operation by configuring the following control bits:

- RXEN = 1
- MODE<3:0> = 0h through 3h
- UxBRGH:L = desired baud rate
- RXPPS = code for desired input pin
- Input pin ANSEL bit = 0
- ON = 1

All other UART control bits are assumed to be in their default state.

Setting the RXEN bit in the UxCON0 register enables the receiver circuitry of the UART. Setting the MODE<3:0> bits in the UxCON0 register configures the UART for the desired Asynchronous mode. Setting the ON bit in the UxCON1 register enables the UART. The TRIS bit corresponding to the selected RX I/O pin must be set to configure the pin as an input.

**Note:** If the RX function is on an analog pin, the corresponding ANSEL bit must be cleared for the receiver to function.

### 31.2.2.2 Receiving Data

Data is recovered from the bit stream by timing to the center of the bits and sampling the input level. In High-Speed mode, there are four BRG clocks per bit and only one sample is taken per bit. In Normal-Speed mode, there are 16 BRG clocks per bit and three samples are taken per bit.

The receiver data recovery circuit initiates character reception on the falling edge of the Start bit. The Start bit, is always a '0'. The Start bit is qualified in the middle of the bit. In Normal-Speed mode only, the Start bit is also qualified at the leading edge of the bit. The following paragraphs describe the majority detect sampling of Normal-Speed mode.

The falling edge starts the baud rate generator (BRG) clock. The input is sampled at the first and second BRG clocks.

If both samples are high then the falling edge is deemed a glitch and the UART returns to the Start bit detection state without generating an error.

If either sample is low, the data recovery circuit continues counting BRG clocks and takes samples at clock counts 7, 8, and 9. When less than two samples are low, the Start bit is deemed invalid and the data recovery circuit aborts character reception, without generating an error, and resumes looking for the falling edge of the Start bit.

When two or more samples are low, the Start bit is deemed valid and the data recovery continues. After a valid Start bit is detected, the BRG clock counter continues and resets at count 16. This is the beginning of the first data bit.

The data recovery circuit counts BRG clocks from the beginning of the bit and takes samples at clocks 7, 8, and 9. The bit value is determined from the majority of the samples. The resulting '0' or '1' is shifted into the RSR. The BRG clock counter continues and resets at count 16. This sequence repeats until all data bits have been sampled and shifted into the RSR.

After all data bits have been shifted in, the first Stop bit is sampled. Stop bits are always a '1'. If the bit sampling determines that a '0' is in the Stop bit position, the framing error is set for this character. Otherwise, the framing error is cleared for this character. See [Section 31.2.2.4 "Receive Framing Error"](#) for more information on framing errors.

### 31.2.2.3 Receive Interrupts

Immediately after all data bits and the Stop bit have been received, the character in the RSR is transferred to the UART receive FIFO. The UxRXIF interrupt flag in the respective PIR register is set at this time, provided it is not being suppressed.

The UxRXIF is suppressed by any of the following:

- FERIF if FERIE is set
- PERIF if PERIE is set

This suspends DMA transfer of data until software processes the error and reads UxRXB to advance the FIFO beyond the error.

UxRXIF interrupts are enabled by setting all of the following bits:

- UxRXIE, Interrupt Enable bit in the PIE register
- GIE, Global Interrupt Enable bits in the INTCON0 register

The UxRXIF interrupt flag bit will be set when not suppressed and there is an unread character in the FIFO, regardless of the state of interrupt enable bits. Reading the UxRXB register will transfer the top character out of the FIFO and reduce the FIFO contents by one. The UxRXIF interrupt flag bit is read-only, it cannot be set or cleared by software.

## 31.2.2.4 Receive Framing Error

Each character in the receive FIFO buffer has a corresponding framing error flag bit. A framing error indicates that the Stop bit was not seen at the expected time. The framing error flag is accessed via the FERIF bit in the UxERRIR register. The FERIF bit represents the frame status of the top unread character of the receive FIFO. Therefore, the FERIF bit must be read before reading UxRXB.

The FERIF bit is read-only and only applies to the top unread character of the receive FIFO. A framing error (FERIF = 1) does not preclude reception of additional characters. It is neither necessary nor possible to clear the FERIF bit directly. Reading the next character from the FIFO buffer will advance the FIFO to the next character and the next corresponding framing error.

The FERIF bit is cleared when the character at the top of the FIFO does not have a framing error or when all bytes in the receive FIFO have been read. Clearing the ON bit resets the receive FIFO, thereby also clearing the FERIF bit.

A framing error will generate a summary UxERR interrupt when the FERIE bit in the UxERRIE register is set. The summary error is reset when the FERIF bit of the top of the FIFO is '0' or when all FIFO characters have been retrieved.

When FERIE is set, UxRXIF interrupts are suppressed when FERIF is '1'.

## 31.2.2.5 Receiver Parity Modes

Even and odd parity is automatically detected when the MODE<3:0> bits are set to '0011' and '0010', respectively. Parity modes receive eight data bits and one parity bit for a total of nine bits for each character. The PERIF bit in the UxERRIR register represents the parity error of the top unread character of the receive FIFO rather than the parity bit itself. The parity error must be read before reading the UxRXB register advances the FIFO.

A parity error will generate a summary UxERR interrupt when the PERIE bit in the UxERRIE register is set. The summary error is reset when the PERIF bit of the top of the FIFO is '0' or when all FIFO characters have been retrieved.

When PERIE is set, UxRXIF interrupts are suppressed when PERIF is '1'.

## 31.2.2.6 Receive FIFO Overflow

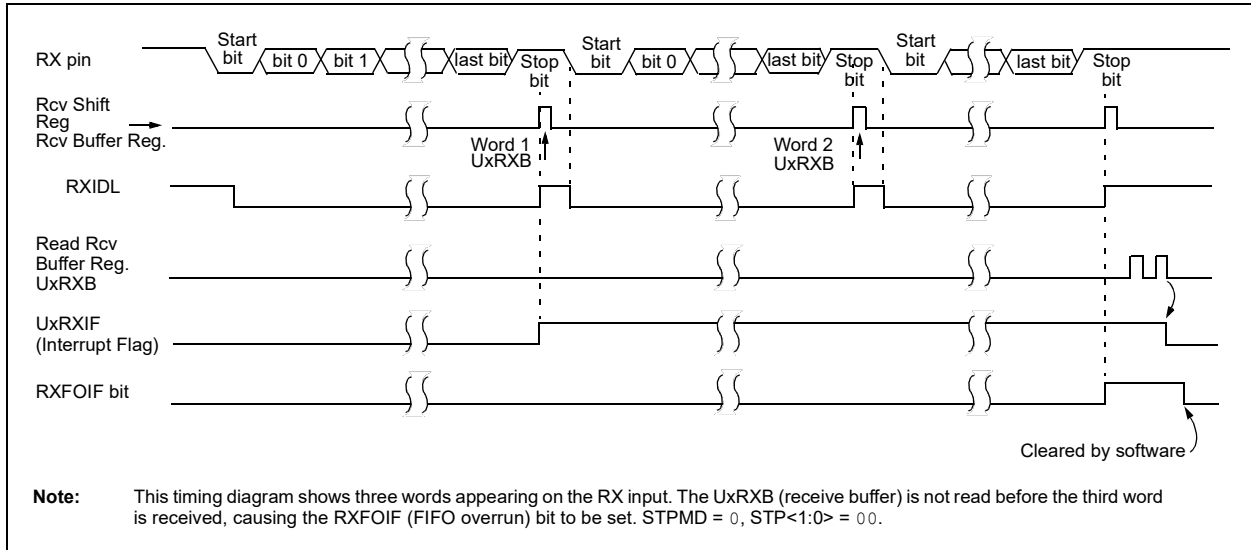
When more characters are received than the receive FIFO can hold, the RXFOIF bit in the UxERRIR register is set. The character causing the overflow condition is discarded. The RUNOVF bit in the UxCON2 register determines how the receive circuit responds to characters while the overflow condition persists. When RUNOVF is set, the receive shifter stays synchronized to the incoming data stream by responding to Start, data, and Stop bits. However, all received bytes not already in the FIFO are discarded. When RUNOVF is cleared, the receive shifter ceases operation and Start, data, and Stop bits are ignored. The receive overflow condition is cleared by reading the UxRXB register and clearing the RXFOIF bit. If the UxRXB register is not read to open a space in the FIFO, the next character received will be discarded and cause another overflow condition.

A receive overflow error will generate a summary UxEIF interrupt when the RXFOIE bit in the UxERRIE register is set.

## 31.2.2.7 Asynchronous Reception Setup

1. Initialize the UxBRGH, UxBRGL register pair and the BRGS bit to achieve the desired baud rate (see [Section 31.17 "UART Baud Rate Generator \(BRG\)"](#)).
2. Configure the RXPPS register for the desired RX pin
3. Clear the ANSEL bit for the RX pin (if applicable).
4. Set the MODE<3:0> bits to the desired Asynchronous mode.
5. Set the RXPOL bit if the data stream is inverted.
6. Enable the serial port by setting the ON bit.
7. If interrupts are desired, set the UxRXIE bit in the PIE<sub>x</sub> register and the GIE bits in the INTCON0 register.
8. Enable reception by setting the RXEN bit.
9. The UxRXIF interrupt flag bit will be set when a character is transferred from the RSR to the receive buffer. An interrupt will be generated if the UxRXIE interrupt enable bit is also set.
10. Read the UxERRIR register to get the error flags.
11. Read the UxRXB register to get the received byte.
12. If an overrun occurred, clear the RXFOIF bit.

**FIGURE 31-5: ASYNCHRONOUS RECEPTION**



## 31.3 Asynchronous Address Mode

A special Address Detection mode is available for use when multiple receivers share the same transmission line, such as in RS-485 systems.

When Asynchronous Address mode is enabled, all data is transmitted and received as 9-bit characters. The 9th bit determines whether the character is an address or data. When the 9th bit is set, the eight Least Significant bits are the address. When the 9th bit is clear, the Least Significant bits are data. In either case, the 9th bit is stored in PERIF when the byte is written to the receive FIFO. When PERIE is also set, the RXIF will be suppressed, thereby suspending DMA transfers allowing software to process the received address.

An address character will enable all receivers that match the address and disable all other receivers. Once a receiver is enabled, all non-address characters will be received until an address character is received that does not match.

### 31.3.1 ADDRESS MODE TRANSMIT

The UART transmitter is enabled for asynchronous address operation by configuring the following control bits:

- TXEN = 1
- MODE<3:0> = 0100
- UxBRGH:L = desired baud rate
- RxyPPS = code for desired output pin
- ON = 1

Addresses are sent by writing to the UxP1L register. This transmits the written byte with the 9th bit set, which indicates that the byte is an address.

Data is sent by writing to the UxTXB register. This transmits the written byte with the 9th bit cleared, which indicates that the byte is data.

To send data to a particular device on the transmission bus, first transmit the address of the intended device. All subsequent data will be accepted only by that device until an address of another device is transmitted.

Writes to UxP1L take precedence over writes to UxTXB. When both the UxP1L and UxTXB registers are written while the TSR is busy, the next byte to be transmitted will be from UxP1L.

To ensure that all data intended for one device is sent before the address is changed, wait until the TXMTIF bit is high before writing UxP1L with the new address.

### 31.3.2 ADDRESS MODE RECEIVE

The UART receiver is enabled for asynchronous address operation by configuring the following control bits:

- RXEN = 1
- MODE<3:0> = 0100
- UxBRGH:L = desired baud rate
- RXPPS = code for desired input pin
- Input pin ANSEL bit = 0
- UxP2L = receiver address
- UxP3L = address mask
- ON = 1

In Address mode, no data will be transferred to the input FIFO until a valid address is received. This is the default state. Any of the following conditions will cause the UART to revert to the default state:

- ON = 0
- RXEN = 0
- Received address does not match

When a character with the 9th bit set is received, the Least Significant eight bits of that character will be qualified by the values in the UxP2L and UxP3L registers.

The byte is XOR'd with UxP2L then AND'd with UxP3L. A match occurs when the result is 0h, in which case, the unaltered received character is stored in the receive FIFO, thereby setting the UxRXIF interrupt bit. The 9th bit is stored in the corresponding PERIF bit, identifying this byte as an address.

An address match also enables the receiver for all data such that all subsequent characters without the 9th bit set will be stored in the receive FIFO.

When the 9th bit is set and a match does not occur, the character is not stored in the receive FIFO and all subsequent data is ignored.

The UxP3L register mask allows a range of addresses to be accepted. Software can then determine the sub-address of the range by processing the received address character.

## 31.4 DMX Mode

DMX is a protocol used in stage and show equipment. This includes lighting, fog machines, motors, etc. The protocol consists of a controller that sends out commands, and receiver such as theater lights that receive these commands. DMX protocol is usually unidirectional, but can be a bidirectional protocol in either Half or Full-Duplex modes. An example of Half-Duplex mode is the RDM (Remote Device Management) protocol that sits on DMX512A. The controller transmits commands and the receiver receives them. Also there are no error conditions or retransmit mechanisms.

DMX, or DMX512A as it is known, consists of a "Universe" of 512 channels. This means that one controller can output up to 512 bytes on a single DMX link. Each equipment on the line is programmed to listen to a consecutive sequence of one or more of these bytes.

For example, a fog machine connected to one of the universes may be programmed to receive one byte, starting at byte number 10, and a lighting unit may be programmed to receive four bytes starting at byte number 22.

### 31.4.1 DMX CONTROLLER

DMX Controller mode is configured with the following settings:

- $\text{MODE}\langle 3:0 \rangle = 1010$
- $\text{TXEN} = 1$
- $\text{RXEN} = 0$
- $\text{TXPOL} = 1$
- $\text{UxP1}$  = One less than the number of bytes to transmit (excluding the Start code)
- $\text{UxBRGH:L}$  = Value to achieve 250K baud rate
- $\text{STP}\langle 1:0 \rangle = 10$  for 2 Stop bits
- $\text{RxyPPS}$  = TX pin output code
- $\text{ON} = 1$

Each DMX transmission begins with a Break followed by a byte called the 'Start Code'. The width of the BREAK is fixed at 25 bit times. The Break is followed by a "Mark After Break" (MAB) Idle period. After this Idle period, the 1st through 'n'th byte is transmitted, where 'n-1' is the value in  $\text{UxP1}$ . See [Figure 31-6](#).

Software sends the Start Code and the 'n' data bytes by writing the  $\text{UxTXB}$  register with each byte to be sent in the desired order. A  $\text{UxTXIF}$  value of '1' indicates when the  $\text{UxTXB}$  is ready to accept the next byte.

The internal byte counter is not accessible to software. Software needs to keep track of the number of bytes written to  $\text{UxTXB}$  to ensure that no more and no less than 'n' bytes are sent because the DMX state machine will automatically insert a Break and reset its internal counter after 'n' bytes are written. One way to ensure synchronization between hardware and software is to

toggle  $\text{TXEN}$  after the last byte of the universe is completely free of the transmit shift register as indicated by the  $\text{TXMTIF}$  bit.

### 31.4.2 DMX RECEIVER

DMX Receiver mode is configured with the following settings:

- $\text{MODE}\langle 3:0 \rangle = 1010$
- $\text{TXEN} = 0$
- $\text{RXEN} = 1$
- $\text{RXPOL} = 1$
- $\text{UxP2}$  = number of first byte to receive
- $\text{UxP3}$  = number of last byte to receive
- $\text{UxBRGH:L}$  = Value to achieve 250K baud rate
- $\text{STP}\langle 1:0 \rangle = 10$  for 2 Stop bits
- $\text{ON} = 1$
- $\text{UxRXPPS}$  = code for desired input pin
- Input pin  $\text{ANSEL}$  bit = 0

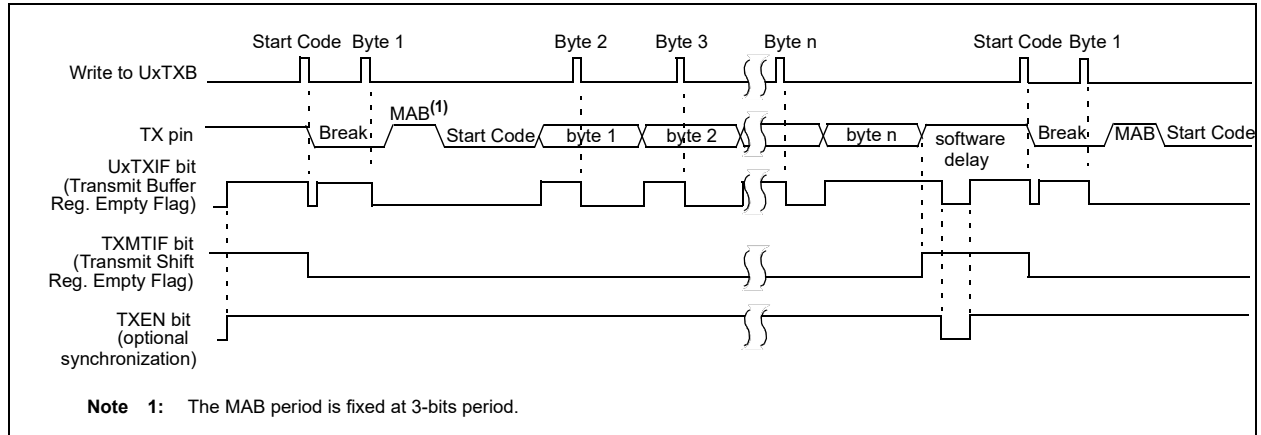
When configured as DMX Receiver, the UART listens for a Break character that is at least 23 bit periods wide. If the Break is shorter than 23 bit times, the Break is ignored and the DMX state machine remains in Idle mode. Upon receiving the Break, the DMX counters will be reset to align with the incoming data stream. Immediately after the Break, the UART will see the "Mark after Break" (MAB). This space is ignored by the UART. The Start Code follows the MAB and will always be stored in the receive FIFO.

After the Start Code, the 1st through 512th byte will be received, but not all of them are stored in the receive FIFO. The UART ignores all received bytes until the ones of interest are received. This is done using the  $\text{UxP2}$  and  $\text{UxP3}$  registers. The  $\text{UxP2}$  register holds the value of the byte number to start the receive process and the  $\text{UxP3}$  register holds the value of the byte number to end the receive process. The byte counter starts at 0 for the first byte after the Start Code. For example, to receive four bytes starting at the 10th byte after the Start Code, write 009h (9 decimal) to  $\text{UxP2H:L}$  and 00Ch (12 decimal) to  $\text{UxP3H:L}$ . The receive FIFO is only 2 bytes deep, therefore the bytes must be retrieved by reading  $\text{UxRXB}$  as they come in to avoid a receive FIFO overrun condition.

Typically two Stop bits are inserted between bytes. If either Stop bit is detected as a '0' then the framing error for that byte will be set.

Since the DMX sequence always starts with a Break, the software can verify that it is in sync with the sequence by monitoring the  $\text{RXBKIF}$  flag to ensure that the next byte received after the  $\text{RXBKIF}$  is processed as the Start Code and subsequent bytes are processed as the expected data.

**FIGURE 31-6: DMX TRANSMIT SEQUENCE**



## 31.5 LIN Modes

LIN is a protocol used primarily in automotive applications. The LIN network consists of two kinds of software processes: a Master process and a Slave process. Each network has only one Master process and one or more Slave processes.

From a physical layer point of view, the UART on one processor may be driven by both a Master and a Slave process, as long as only one Master process exists on the network.

A LIN transaction consists of a Master process followed by a Slave process. The Slave process may involve more than one slave where one is transmitting and the other(s) are receiving. The transaction begins by the following Master process transmission sequence:

1. Break
2. Delimiter bit
3. Sync Field
4. PID byte

The PID determines which Slave processes are expected to respond to the Master. When the PID byte is complete, the TX output remains in the Idle state. One or more of the Slave processes may respond to the Master process. If no one responds within the inter-byte period, the Master is free to start another transmission. The inter-byte period is timed by software using a means other than the UART.

The Slave process follows the Master process. When the slave software recognizes the PID then that Slave process responds by either transmitting the required response or by receiving the transmitted data. Only Slave processes send data. Therefore, Slave processes receiving data are receiving that of another Slave process.

When a slave sends data, the slave UART automatically calculates the checksum for the transmitted bytes as they are sent and appends the inverted checksum byte to the slave response.

When a slave receives data, the checksum is accumulated on each byte as it is received using the same algorithm as the sending process. The last byte, which is the inverted checksum value calculated by the sending process, is added to the locally calculated checksum by the UART. The check passes when the result is all '1's, otherwise the check fails and the CERIF bit is set.

Two methods for computing the checksum are available: legacy and enhanced. The legacy checksum includes only the data bytes. The enhanced checksum includes the PID and the data. The C0EN control bit in the UxCON2 register determines the checksum method. Setting C0EN to '1' selects the enhanced method. Software must select the appropriate method before the Start bit of the checksum byte is received.

### 31.5.1 LIN MASTER/SLAVE MODE

The LIN Master mode includes capabilities to generate Slave processes. The Master process stops at the PID transmission. Any data that is transmitted in Master/Slave mode is done as a Slave process. LIN Master/Slave mode is configured by the following settings:

- MODE<3:0> = 1100
- TXEN = 1
- RXEN = 1
- UxBRGH:L = Value to achieve desired baud rate
- TXPOL = 0 (for high Idle state)
- STP = desired Stop bits selection
- C0EN = desired checksum mode
- RxyPPS = TX pin selection code
- TX pin TRIS control = 0
- ON = 1

**Note:** The TXEN bit must be set before the Master process is received and remain set while in LIN mode whether or not the Slave process is a transmitter.

The Master process is started by writing the PID to the UxP1L register when UxP2 is '0' and the UART is Idle. The UxTXIF will not be set in this case. Only the six Least Significant bits of UxP1L are used in the PID transmission.

The two Most Significant bits of the transmitted PID are PID parity bits. PID<6> is the exclusive-or of PID bits 0,1,2, and 4. PID<7> is the inverse of the exclusive-or of PID bits 1,3,4, and 5.

The UART calculates and inserts these bits in the serial stream.

Writing UxP1L automatically clears the UxTXCHK and UxRXCHK registers and generates the Break, delimiter bit, Sync character (55h), and PID transmission portion of the transaction. The data portion of the transaction that follows, if there is one, is a Slave process. See [Section 31.5.2 "LIN Slave Mode"](#) for more details of that process. The master receives its own PID when RXEN is set. Software performs the Slave process corresponding to the PID that was sent and received. Attempting to write UxP1L before an active master process is complete will not succeed. Instead, the TXWRE bit will be set.

## 31.5.2 LIN SLAVE MODE

LIN Slave mode is configured by the following settings:

- MODE<3:0> = 1011
- TXEN = 1
- RXEN = 1
- UxP2 = Number of data bytes to transmit
- UxP3 = Number of data bytes to receive
- UxBRGH:L = Value to achieve default baud rate
- TXPOL = 0 (for high Idle state)
- STP = desired Stop bits selection
- C0EN = desired checksum mode
- RxyPPS = TX pin selection code
- TX pin TRIS control = 0
- ON = 1

The Slave process starts upon detecting a Break on the RX pin. The Break clears the UxTXCHK, UxRXCHK, UxP2, and UxP3 registers. At the end of the Break, the auto-baud circuitry is activated and the baud rate is automatically set using the Sync character following the Break. The character following the Sync character is received as the PID code and is saved in the receive FIFO. The UART computes the two PID parity bits from the six Least Significant bits of the PID. If either parity bit does not match the corresponding bit of the received PID code, the PERIF flag is set and saved at the same FIFO location as the PID code. The UxRXIF bit is set indicating that the PID is available.

Software retrieves the PID by reading the UxRXB register and determines the Slave process to execute from that. The checksum method, number of data bytes, and whether to send or receive data, is defined by software according to the PID code.

### 31.5.2.1 LIN Slave Receiver

When the Slave process is a receiver, the software performs the following tasks:

- UxP3 register is written with a value equal to the number of data bytes to receive.
- C0EN bit is set or cleared to select the appropriate checksum. This must be completed before the Start bit of the checksum byte is received.
- Each byte of the process response is read from UxRXB when UxRXIF is set.

The UART updates the checksum on each received byte. When the last data byte is received, the computed checksum total is stored in the UxRXCHK register. The next received byte is saved in the receive FIFO and added with the value in UxRXCHK. The result of this addition is not accessible. However, if the result is not all '1's, the CERIF bit in the UxERRIR is set. The CERIF flag persists until cleared by software. Software needs to read UxRXB to remove the checksum byte from the FIFO, but the byte can be discarded if not needed for any other purpose.

After the checksum is received, the UART ignores all activity on the RX pin until a Break starts the next transaction.

### 31.5.2.2 LIN Slave Transmitter

When the Slave process is a transmitter, software performs the following tasks in the order shown:

- UxP2 register is written with a value equal to the number of bytes to transmit. This will enable TXIF flag which is disabled when UxP2 is '0'.
- C0EN bit is set or cleared to select the appropriate checksum
- Inter-byte delay is performed
- Each byte of the process response is written to UxTXB when UxTXIF is set

The UART accumulates the checksum as each byte is written to UxTXB. After the last byte is written, the UART stores the calculated checksum in the UxTXCHK register and transmits the inverted result as the last byte in the response.

The TXIF flag is disabled when UxP2 bytes have been written. Any writes to UxTXB that exceed the UxP2 count will be ignored and set the TXWRE flag in the UxFIFO register.



## 31.6 DALI Mode

DALI is a protocol used for intelligent lighting control for building automation. The protocol consists of 'Control Devices' and 'Control Gear'. A Control Device is an application controller that sends out commands to the light fixtures. The light fixture itself is termed as a control gear. The communication is done using Manchester encoding, which is performed by the UART hardware.

Manchester encoding consists of the clock and data in a single bit stream. A high-to-low or a low-to-high transition always occurs in the middle of the bit period and is not guaranteed to occur at the bit period boundaries.

When the consecutive bits in the bit stream are of the same value (i.e., consecutive '1's or consecutive '0's), a transition occurs at the bit boundary. However, when the bit value changes, there is no transition at the bit boundary. According to the standard, a half-bit time is typically 416.7  $\mu$ s long. A double half-bit time or a single bit is typically 833.3  $\mu$ s.

The protocol is inherently half-duplex. Communication over the bus occurs in the form of forward and backward frames. Wait times between the frames are defined in the standard to prevent collision between the frames.

A Control Device transmission is termed as the 'Forward Frame'. In the DALI 2.0 standard, a forward frame can be two or three bytes in length. The two-byte forward frame is used for communication between control device and control gear, whereas the three-byte forward frame is used for communication between Control Devices on the bus. The first byte in the forward frame is the control byte and is followed by either one or two data bytes. The transaction begins when the Control Device starts a transmission. Unlike other protocols, each byte in the frame is transmitted MSB first. Typical frame timing is as shown in [Figure 31-8](#).

During communication between two control devices, three bytes are required to be transmitted. In this case, the software must write the third byte to UxTXB as soon as UxTXIF goes True and before the output shifter becomes empty. This ensures that the three bytes of the forward frame are transmitted back-to-back, without any interruption.

All control gear on the bus receive the forward frame. If the forward frame requires a reply to be sent, one of the control gear may respond with a single byte, called the 'Backward Frame'. The 2.0 standard requires the control gear to begin transmission of the backward frame between 5.5 ms to 10.5 ms (~14 to 22 half-bit times) after reception of the forward frame. Once the backward frame is received by the Control Device, it is required to wait a minimum of 2.4 ms (~6 half-bit times). After this wait time, the Control Device is free to transmit another forward frame (see [Figure 31-9](#)).

A Start bit is used to indicate the start of the forward and backward frames. When ABDEN = 0, the receiver bit rate is determined by the BRG register. When ABDEN = 1, the first bit synchronizes the receiver with the transmitter and sets the receiver bit rate. The low period of the Start bit is measured and is used as the timing reference for all data bits in the forward and backward frames. The ABDOVF bit is set if the Start bit low period causes the measurement counter to overflow. All bits following the Start bit are data bits. The bit stream terminates when no transition is detected in the middle of a bit period (see [Figure 31-7](#)).

Forward and backward frames are terminated by two Idle bit periods or Stop bits. Normally, these start in the first bit period of a byte. If both Stop bits are valid, the byte reception is terminated and the CERIF bit in the UxERRIR1 register is set. This bit needs to be cleared in software.

If either of the Stop bits is invalid, the frame is tagged as invalid by saving it as a null byte and setting the framing error in the receive FIFO.

A framing error also occurs when no transition is detected on the bus in the middle of a bit period when the byte reception is not complete. In such a scenario, the byte will be saved with the FERIF bit.

### 31.6.1 CONTROL DEVICE

Control Device mode is configured with the following settings:

- MODE<3:0> = 1000
- TXEN = 1
- RXEN = 1
- UxP1 = Forward frames are held for transmission this number of half-bit periods after the completion of a forward or backward frame.
- UxP2 = Forward/backward frame threshold delimiter. Any reception that starts this number of half bit periods after the completion of a forward or backward frame is detected as forward frame and sets the PERIF flag of the corresponding received byte.
- UxBRGH:L = Value to achieve 1200 baud rate
- TXPOL = appropriate polarity for interface circuit
- STP<1:0> = 10 for two Stop bits
- RxyPPS = TX pin selection code
- TX pin TRIS control = 0
- ON = 1

A forward frame is initiated by writing the control byte to the UxTXB register. Each data byte after the control byte must be written to the UxTXB register as soon as UxTXIF goes true. It is necessary to perform every write after UxTXIF goes true to ensure the transmit buffer is ready to accept the byte. Each write must also occur before the TXMTIF bit goes true, to ensure that the bit stream of forward frame is generated without interruption.



When TXMTIF goes true, indicating the transmit shift register has completed sending the last byte in the frame, the TX output is held in the Idle state for the number of half-bit periods selected by the STP bits in the UxCON2 register.

After the last Stop bit, the TX output is held in the Idle state for an additional wait time determined by the half-bit period count in the UxP1 register. For example, a 2450  $\mu$ s delay (~6 half-bit times) requires a value of 6 in UxP1L.

Any writes to the UxTXB register that occur after TXMTIF goes true, but before the UxP1 wait time, will be held and then transmitted immediately following the wait time. If a backward frame is received during the wait time, any bytes that may have been written to UxTXB will be transmitted after completion of the backward frame reception the backward frame plus the UxP1 wait time.

The wait timer is reset by the backward frame and starts over immediately following the Stop bits of the backward frame. Data pending in the transmit shift register will be sent when the wait time elapses.

To replace or delete any pending forward frame data, the TXBE bit needs to be set to flush the shift register and transmit buffer, then write the new control byte to the UxTXB register. The new control byte will be held in the buffer and sent as the beginning of the next forward frame following the UxP1 wait time.

In Control Device mode, PERIF is set when a forward frame is received. This helps the software distinguish whether the received byte is part of a forward frame from a Control Device (either from the Control Device under consideration or from another Control Device on the bus) or a backward frame from a Control Gear.

## 31.6.2 CONTROL GEAR

The Control Gear mode is configured with the following settings:

- $\text{MODE}\langle 3:0 \rangle = 1001$
- $\text{TXEN} = 1$
- $\text{RXEN} = 1$
- $\text{UxP1}$  = Backward frames are held for transmission this number of half-bit periods after the completion of a forward frame.
- $\text{UxP2}$  = Forward/backward frame threshold delimiter. Idle periods more than this number of half-bit periods are detected as forward frames.
- $\text{UxBRGH:L}$  = Value to achieve 1200 baud rate
- $\text{TXPOL}$  = appropriate polarity for interface circuit
- $\text{RXPOL}$  = same as  $\text{TXPOL}$
- $\text{STP} = 10$  for two Stop bits
- $\text{RxyPPS}$  = TX pin output code
- TX pin TRIS control = 0
- $\text{RXPPS}$  = RX pin selection code
- RX pin TRIS control = 1
- Input pin ANSEL bit = 0
- $\text{ON} = 1$

The UART starts listening for a forward frame when the Control Gear mode is entered. Only the frames that follow an Idle period longer than  $\text{UxP2}$  half-bit periods are detected as forward frames. Backward frames from other Control Gear are ignored. Only forward frames will be stored in UxRXB. This is necessary because a backward frame can be sent only as a response to a forward frame.

The forward frame is received one byte at a time in the receive FIFO and retrieved by reading the UxRXB register. The end of the forward frame starts a timer to delay the backward frame response by wait time equal to the number of half-bit periods stored in  $\text{UxP1}$ . The data received in the forward frame is processed by the application software. If the application decides to send a backward frame in response to the forward frame, the value of the backward frame is written to UxTXB. This value is held for transmission in the transmit shift register until the wait time expires and is then transmitted.

If the backward frame data is written to UxTXB after the wait time has expired, it is held in the UxTXB register until the end of the wait time following the next forward frame. The TXMTIF bit is false when the backward frame data is held in the transmit shift register. Receiving a UxRXIF interrupt before the TXMTIF goes true indicates that the backward frame write was too late and another forward frame was received before sending the backward frame. The pending backward frame has to be flushed by setting the TXBE bit, to prevent it from being sent after the next Forward Frame.

**FIGURE 31-7: MANCHESTER TIMING**

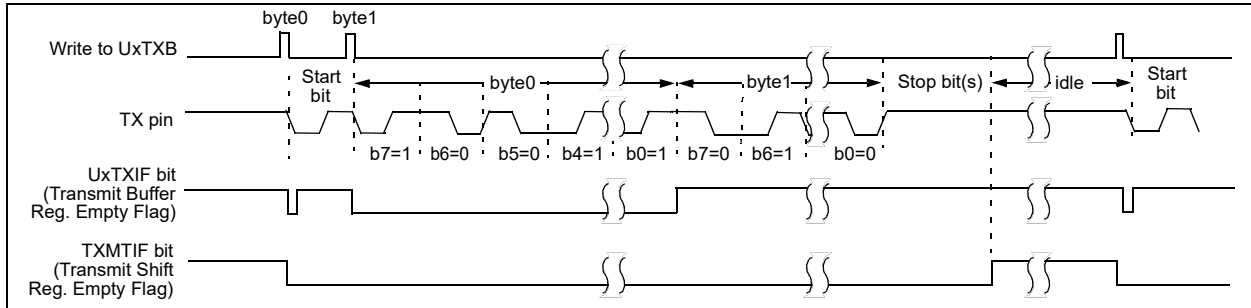


**FIGURE 31-8: DALI FRAME TIMING**

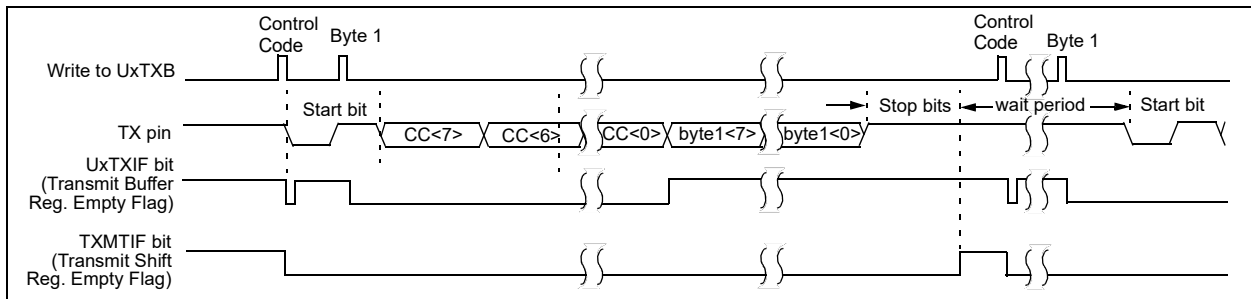
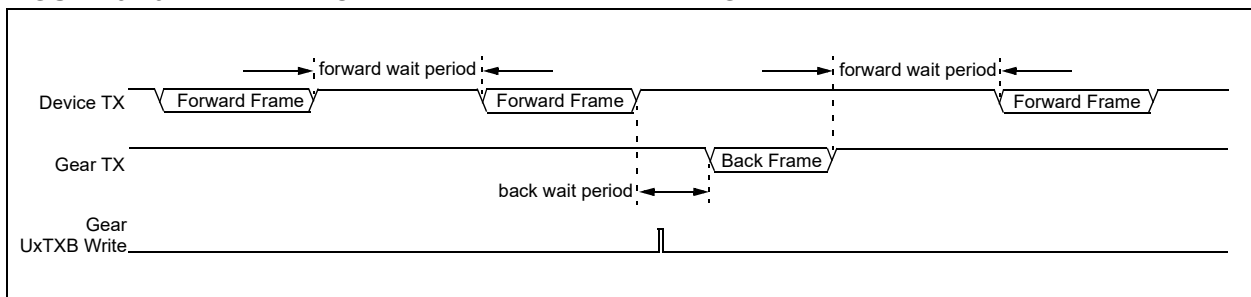


**FIGURE 31-9: DALI FORWARD/BACK FRAME TIMING**



## 31.7 General Purpose Manchester

General purpose Manchester is a subset of the DALI mode. When the UxP1L register is cleared, there is no minimum wait time between frames. This allows full and half-duplex operation because writes to the UxTXB are not held waiting for a receive operation to complete.

General purpose Manchester operation maintains all other aspects of DALI mode such as:

- Single-pulse Start bit
- Most Significant bit first
- No stop periods between back-to-back bytes

General purpose Manchester mode is configured with the following settings:

- $MODE<3:0> = 1000$
- $TXEN = 1$
- $RXEN = 1$
- $UxP1 = 0h$
- $UxBRGH:L = \text{desired baud rate}$
- $TXPOL$  and  $RXPOL = \text{desired Idle state}$
- $STP = \text{desired number of stop periods}$
- $RxyPPS = \text{TX pin selection code}$
- TX pin TRIS control = 0
- $RXPPS = \text{RX pin selection code}$
- RX pin TRIS control = 1
- Input pin ANSEL bit = 0
- $ON = 1$

The Manchester bit stream timing is shown in [Figure 31-7](#).

## 31.8 Polarity

Receive and transmit polarity is user selectable and affects all modes of operation.

The idle level is programmable with the polarity control bits in the UxCON2 register. The control bits default to '0', which select a high idle level. The low level Idle state is selected by setting the control bit to '1'. TXPOL controls the TX idle level. RXPOL controls the RX idle level.

## 31.9 Stop Bits

The number of Stop bits is user selectable with the STP bits in the UxCON2 register. The STP bits affect all modes of operation.

Stop bits selections include:

- 1 transmit with receive verify on first
- 1.5 transmit with receive verify on first
- 2 transmit with receive verify on both
- 2 transmit with receive verify on first only

In all modes, except DALI, the transmitter is idle for the number of Stop bit periods between each consecutively transmitted word. In DALI, the Stop bits are generated after the last bit in the transmitted data stream.

The input is checked for the idle level in the middle of the first Stop bit, when receive verify on first is selected, as well as in the middle of the second Stop bit, when verify on both is selected. If any Stop bit verification indicates a non-idle level, the framing error FERIF bit is set for the received word.

### 31.9.1 DELAYED UXRIF

When operating in Half-Duplex mode, where the microcontroller needs to reverse the transceiver direction after a reception, it may be more convenient to hold off the UxRXIF interrupt until the end of the Stop bits to avoid line contention. The user selects when the UxRXIF interrupt occurs with the STPMD bit in the UxFIFO register. When STPMD is '1', the UxRXIF occurs at the end of the last Stop bit. When STPMD is '0', UxRXIF occurs when the received byte is stored in the receive FIFO. When  $STP<1:0> = 10$ , the store operation is performed in the middle of the second Stop bit, otherwise, it is performed in the middle of the first Stop bit. The FERIF and PERIF interrupts are not delayed with STPMD. Only UxRXIF is delayed when STPMD is set and should be the only indicator for reversing transceiver direction.

## 31.10 Operation after FIFO overflow

The Receive Shift Register (RSR) can be configured to stop or continue running during a receive FIFO overflow condition. Stopped operation is the Legacy mode.

When the RSR continues to run during an overflow condition, the first word received after clearing the overflow will always be valid.

When the RSR is stopped during an overflow condition, synchronization with the Start bits is lost. Therefore, the first word received after the overflow is cleared may start in the middle of a word.

Operation during overflow is selected with the RUNOVF bit in the UxCON2 register. Setting the RUNOVF bit selects the run during overflow method.

## 31.11 Receive and Transmit Buffers

The UART uses small buffer areas to transmit and receive data. These are sometimes referred to as FIFOs.

The receiver has a Receive Shift Register (RSR) and two buffer registers. The buffer at the top of the FIFO (earliest byte to enter the FIFO) is by retrieved by reading the UxRXB register.

The transmitter has one Transmit Shift Register (TSR) and one buffer register. Writes to UxTXB go to the transmit buffer then immediately to the TSR, if it is empty. When the TSR is not empty, writes to UxTXB are held then transferred to the TSR when it becomes available.

### 31.11.1 FIFO STATUS

The UxFIFO register contains several Status bits for determining the state of the receive and transmit buffers.

The RXBE bit indicates that the receive FIFO is empty. This bit is essentially the inverse of UxRXIF. The RXBF bit indicates that the receive FIFO is full.

The transmitter has only one buffer register so the Status bits are essentially a copy and inverse of the UxTXIF bit. The TXBE bit indicates that the buffer is empty (same as UxTXIF) and the TXBF bit indicates that the buffer is full (UxTXIF inverse). A third transmitter Status bit, TXWRE (transmit write error), is set whenever a UxTXB write is performed when the TXBF bit is set. This indicates that the write was unsuccessful.

### 31.11.2 FIFO RESET

All modes support resetting the receive and transmit buffers.

The receive buffer is flushed and all unread data discarded when the RXBE bit in the UxFIFO register is written to '1'. The MOVWF instruction with the TXBE bit cleared should be used to avoid inadvertently clearing a byte pending in the TSR when UxTXB is empty.

Data written to UxTXB when TXEN is low will be held in the Transmit Shift Register (TSR) then sent when TXEN is set. The transmit buffer and inactive TSR are flushed by setting the TXBE bit in the UxFIFO register. Setting TXBE while a character is actively transmitting from the TSR will complete the transmission without being flushed.

Clearing the ON bit will discard all received data and transmit data pending in the TSR and UxTXB.

## 31.12 Flow Control

This section does not apply to the LIN, DALI, or DMX modes.

Flow control is the means by which a sending UART data stream can be suspended by a receiving UART. Flow control prevents input buffers from overflowing without software intervention. The UART supports both hardware and XON/XOFF methods of flow control.

The flow control method is selected with the FLO<1:0> bits in the UxCON2 register. Flow control is disabled when are both bits are cleared.

### 31.12.1 HARDWARE FLOW CONTROL

Hardware flow control is selected by setting the FLO<1:0> bits to '10'.

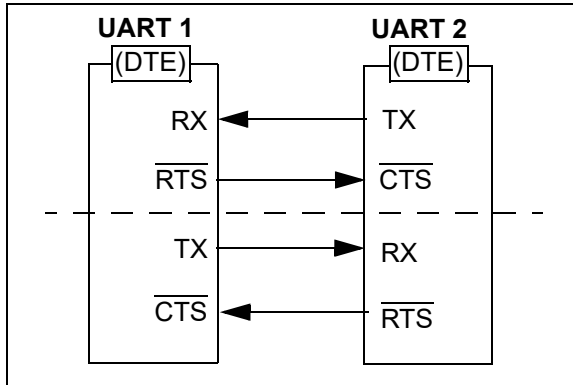
Hardware flow control consists of three lines. The RS-232 signal names for two of these are  $\overline{\text{RTS}}$ , and  $\overline{\text{CTS}}$ . Both are low true. The third line may be used to control an RS-485 transceiver. The signal name for this is TXDE for transmit drive enable. This output is high when the TX output is actively sending a character and low at all other times. The UART is configured as DTE (computer) equipment which means  $\overline{\text{RTS}}$  is an output and  $\overline{\text{CTS}}$  is an input.

The  $\overline{\text{RTS}}$  and  $\overline{\text{CTS}}$  signals work as a pair to control the transmission flow. A DTE-to-DTE configuration connects the  $\overline{\text{RTS}}$  output of the receiving UART to the  $\overline{\text{CTS}}$  input of the sending UART. Refer to [Figure 31-10](#).

The UART receiving data asserts the  $\overline{\text{RTS}}$  output low when the input FIFO is empty. When a character is received, the  $\overline{\text{RTS}}$  output goes high until the UxRXB is read to free up both FIFO locations.

When the  $\overline{\text{CTS}}$  input goes high after a byte has started to transmit, the transmission will complete normally. The receiver accommodates this by accepting the character in the second FIFO location even when the  $\overline{\text{CTS}}$  input is high.

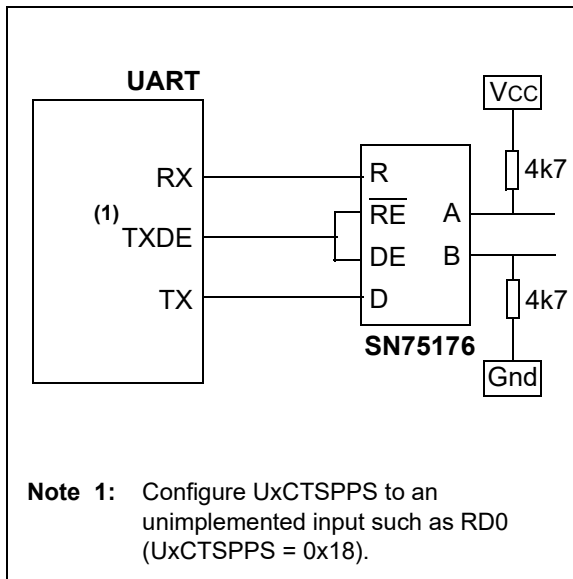
**FIGURE 31-10: FLOW CONTROL**



## 31.12.2 RS-485 TRANSCEIVER CONTROL

Hardware flow control can be used to control the direction of an RS-485 transceiver as shown in [Figure 31-11](#). Configure the CTS input to be always enabled by setting the UxCTSPPS selection to an unimplemented port pin such as RD0. When the signal and control lines are configured as shown in [Figure 31-11](#), then the UART will not receive its own transmissions. To verify that there are no collisions on the RS-485 lines then the transceiver RE control can be disconnected from TXDE and tied low thereby enabling loop-back reception of all transmissions. See [Section 31.14 “Collision Detection”](#) for more information.

**FIGURE 31-11: RS-485 CONFIGURATION**



## 31.12.3 XON/XOFF FLOW CONTROL

XON/XOFF flow control is selected by setting the FLO<1:0> bits to '01'.

XON/XOFF is a data based flow control method. The signals to suspend and resume transmission are special characters sent by the receiver to the transmitter. The advantage is that additional hardware lines are not needed.

XON/XOFF flow control requires full-duplex operation because the transmitter must be able to receive the signal to suspend transmitting while the transmission is in progress. Although XON and XOFF are not defined in the ASCII code, the generally accepted values are 13h for XOFF and 11h for XON. The UART uses those codes.

The transmitter defaults to XON, or transmitter enabled. This state is also indicated by the read-only XON bit in the UxFIFO register.

When an XOFF character is received, the transmitter stops transmitting after completing the character actively being transmitted. The transmitter remains disabled until an XON character is received.

XON will be forced on when software toggles the TXEN bit.

When the RUNOVF bit in the UxCON2 register is set then XON and XOFF characters continue to be received and processed without the need to clear the input FIFO by reading the UxRXB. However, if the RUNOVF bit is clear then the UxRXB must be read to avoid a receive overflow which will suspend flow control when the receive buffer overflows.

## 31.13 Checksum

This section does not apply to the LIN mode, which handles checksums automatically.

The transmit and receive checksum adders are enabled when the C0EN bit in the UxCON2 register is set. When enabled, the adders accumulate every byte that is transmitted or received. The accumulated sum includes the carry of the addition. Software is responsible for clearing the checksum registers before a transaction and performing the check at the end of the transaction.

The following is an example of how the checksum registers could be used in the Asynchronous modes.

### 31.13.1 TRANSMIT CHECKSUM METHOD

1. Clear the UxTXCHK register.
2. Set the C0EN bit.
3. Send all bytes of the transaction output.
4. Invert UxTXCHK and send the result as the last byte of the transaction.

### 31.13.2 RECEIVE CHECKSUM METHOD

1. Clear the UxRXCHK register.
2. Set the C0EN bit.
3. Receive all bytes in the transaction including the checksum byte.
4. Set MSb of UxRXCHK if 7-bit mode is selected.
5. Add 1 to UxRXCHK.
6. If the result is '0', the checksum passes, otherwise it fails.

The CERIF checksum interrupt flag is not active in any mode other than LIN.

## 31.14 Collision Detection

External forces that interfere with the transmit line are detected in all modes of operation with collision detection. Collision detection is always active when RXEN and TXEN are both set.

When the receive input is connected to the transmit output through either the same I/O pin or external circuitry, a character will be received for every character transmitted. The collision detection circuit provides a warning when the word received does not match the word transmitted.

The TXCIF flag in the UxERRIR register is used to signal collisions. This signal is only useful when the TX output is looped back to the RX input and everything that is transmitted is expected to be received. If more than one transmitter is active at the same time, it can be assumed that the TX word will not match the RX word. The TXCIF detects this mismatch and flags an interrupt. The TXCIF bit will also be set in DALI mode transmissions when the received bit is missing the expected mid-bit transition.

Collision detection is always active, regardless of whether or not the RX input is connected to the TX output. It is up to the user to disable the TXCIE bit when collision interrupts are not required.

The software overhead of unloading the receive buffer of transmitted data is avoided by setting the RUNOVF bit in UxCON2 and ignoring the receive interrupt and letting the receive buffer overflow. When the transmission is complete, prepare for receiving data by flushing the receive buffer (see [Section 31.11.2, FIFO Reset](#)) and clearing the RXFOIF overflow flag in the UxERRIR register.

## 31.15 RX/TX Activity Timeout

The UART works in conjunction with the HLT timers to monitor activity on the RX and TX lines. Use this feature to determine when there has been no activity on the receive or transmit lines for a user specified period of time.

To use this feature, set the HLT to the desired timeout period by a combination of the HLT clock source, timer prescale value, and timer period registers. Configure the HLT to reset on the UART TX or RX line and start the HLT at the same time the UART is started. UART activity will keep resetting the HLT to prevent a full HLT period from elapsing. When there has been no activity on the selected TX or RX line for longer than the HLT period then an HLT interrupt will occur signaling the timeout event.

For example, the following register settings will configure HLT2 for a 5 ms timeout of no activity on U1RX:

- T2PR = 0x9C (156 prescale periods)
- T2CLKCON = 0x05 (500 kHz internal oscillator)
- T2HLT = 0x04 (free running, reset on rising edge)
- T2RST = 0x15 (reset on U1RX)
- T2CON = 0xC0 (Timer2 on with 1:16 prescale)

## 31.16 Clock Accuracy with Asynchronous Operation

The factory calibrates the internal oscillator block output (INTOSC). However, the INTOSC frequency may drift as VDD or temperature changes, and this directly affects the asynchronous baud rate. Two methods may be used to adjust the baud rate clock, but both require a reference clock source of some kind.

The first (preferred) method uses the OSCTUNE register to adjust the INTOSC output. Adjusting the value of the OSCTUNE register allows for fine resolution changes to the system clock source. See [Section 7.2.2.3 “Internal Oscillator Frequency Adjustment”](#) for more information.

The other method adjusts the value of the Baud Rate Generator. This can be done automatically with the Auto-Baud Detect feature (see [Section 31.17.1 “Auto-Baud Detect”](#)). There may not be fine enough resolution when adjusting the Baud Rate Generator to compensate for a gradual change of the peripheral clock frequency.

## 31.17 UART Baud Rate Generator (BRG)

The Baud Rate Generator (BRG) is a 16-bit timer that is dedicated to the support of the UART operation.

The UxBRGH, UxBRGL register pair determines the period of the free running baud rate timer. The multiplier of the baud rate period is determined by the BRGS bit in the UxCON0 register.

[Table 31-1](#) contains the formulas for determining the baud rate. [Example 31-1](#) provides a sample calculation for determining the baud rate and baud rate error.

The high baud rate range (BRGS = 1) is intended to extend the baud rate range up to a faster rate when the desired baud rate is not possible otherwise. Using the normal baud rate range (BRGS = 0) is recommended when the desired baud rate is achievable with either range.

Writing a new value to the UxBRGH, UxBRGL register pair causes the BRG timer to be reset (or cleared). This ensures that the BRG does not wait for a timer overflow before outputting the new baud rate.

If the system clock is changed during an active receive operation, a receive error or data loss may result. To avoid this problem, check the status of the RXIDL bit to make sure that the receive operation is idle before changing the system clock.

### EXAMPLE 31-1: CALCULATING BAUD RATE ERROR

For a device with FOSC of 16 MHz, desired baud rate of 9600, Asynchronous mode, BRGS = 0:

$$\text{Desired Baud Rate} = \frac{F_{\text{OSC}}}{16([UxBRG] + 1)}$$

$$X = \frac{\frac{F_{\text{OSC}}}{\text{Desired Baud Rate}}}{16} - 1$$

$$= \frac{\frac{16000000}{9600}}{16} - 1$$

$$= [103.17] = 103$$

$$\text{Calculated Baud Rate} = \frac{16000000}{16(103 + 1)}$$

$$= 9615$$

$$\text{Error} = \frac{\text{Calc. Baud Rate} - \text{Desired Baud Rate}}{\text{Desired Baud Rate}}$$

$$= \frac{(9615 - 9600)}{9600} = 0.16\%$$

**TABLE 31-1: BAUD RATE FORMULAS**

| BRGS | BRG/UART Mode | Baud Rate Formula          |
|------|---------------|----------------------------|
| 1    | High Rate     | $F_{\text{OSC}}/[4(n+1)]$  |
| 0    | Normal Rate   | $F_{\text{OSC}}/[16(n+1)]$ |

**Legend:** n = value of UxBRGH, UxBRGL register pair.

## 31.17.1 AUTO-BAUD DETECT

The UART module supports automatic detection and calibration of the baud rate in the 8-bit Asynchronous and LIN modes. However, setting ABDEN to start auto-baud detection is neither necessary, nor possible in LIN mode because that mode supports auto-baud detection automatically at the beginning of every data packet. Enabling auto-baud detect with the ABDEN bit applies to the Asynchronous modes only.

When Auto-Baud Detect (ABD) is active, the clock to the BRG is reversed. Rather than the BRG clocking the incoming RX signal, the RX signal is timing the BRG. The Baud Rate Generator is used to time the period of a received 55h (ASCII “U”), which is the Sync character for the LIN bus. The unique feature of this character is that it has five falling edges, including the Start bit edge, five rising edges including the Stop bit edge.

In 8-bit Asynchronous mode, setting the ABDEN bit in the UxCON0 register enables the auto-baud calibration sequence. The first falling edge of the RX input after ABDEN is set will start the auto-baud calibration sequence. While the ABD sequence takes place, the UART state machine is held in idle. On the first falling edge of the receive line, the UxBRGH begins counting up using the BRG counter clock as shown in Figure 31-12. The fifth falling edge will occur on the RX pin at the beginning of the bit 7 period. At that time, an accumulated value totaling the proper BRG period is left in the UxBRGH, UxBRGL register pair, the ABDEN bit is automatically cleared and the ABDIF interrupt flag is set. ABDIF must be cleared by software.

RXIDL indicates that the sync input is active. RXIDL will go low on the first falling edge and go high on the fifth rising edge.

The BRG auto-baud clock is determined by the BRGS bit as shown in Table 31-2. During ABD, the internal BRG register is used as a 16-bit counter. However, the UxBRGH and UxBRGL registers retain the previous BRG value until the auto-baud process is successfully completed. While calibrating the baud rate period, the internal BRG register is clocked at 1/8th the BRG base clock rate. The resulting byte measurement is the average bit time when clocked at full speed and is transferred to the UxBRGH and UxBRGL registers when complete.

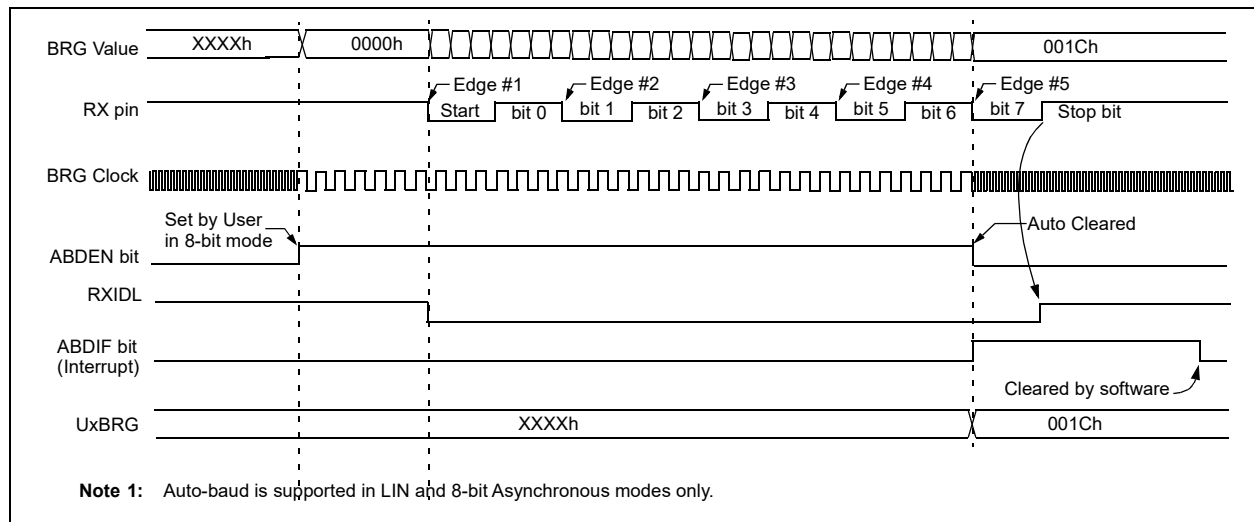
**Note 1:** If the WUE bit is set with the ABDEN bit, auto-baud detection will occur on the byte following the Break character (see Section 31.17.3 “Auto-Wake-up on Break”).

**2:** It is up to the user to determine that the incoming character baud rate is within the range of the selected BRG clock source. Some combinations of oscillator frequency and UART baud rates are not possible.

**TABLE 31-2: BRG COUNTER CLOCK RATES**

| BRGS | BRG Base Clock | BRG ABD Clock |
|------|----------------|---------------|
| 1    | Fosc/4         | Fosc/32       |
| 0    | Fosc/16        | Fosc/128      |

**FIGURE 31-12: AUTOMATIC BAUD RATE CALIBRATION**





## 31.17.2 AUTO-BAUD OVERFLOW

During the course of automatic baud detection, the ABDOVF bit in the UxERRIR register will be set if the baud rate counter overflows before the fifth falling edge is detected on the RX pin. The ABDOVF bit indicates that the counter has exceeded the maximum count that can fit in the 16 bits of the UxBRGH:UxBRGL register pair. After the ABDOVF bit has been set, the state machine continues to search until the fifth falling edge is detected on the RX pin. Upon detecting the fifth falling RX edge, the hardware will set the ABDIF interrupt flag and clear the ABDEN bit in the UxCON0 register. The UxBRGH and UxBRGL register values retain their previous value. The ABDIF flag in the UxUIR register and ABDOVF flag in the UxERRIR register can be cleared by software directly. To generate an interrupt on an auto-baud overflow condition, all the following bits must be set:

- ABDOVE bit in the UxERRIE register
- UxEIE bit in the P1Ex register
- PIE and GIE bits in the INTCON register

To terminate the auto-baud process before the ABDIF flag is set, clear the ABDEN bit, then clear the ABDOVF bit in the UxERRIR register.

## 31.17.3 AUTO-WAKE-UP ON BREAK

During Sleep mode, all clocks to the UART are suspended. Because of this, the Baud Rate Generator is inactive and a proper character reception cannot be performed. The Auto-Wake-up feature allows the controller to wake-up due to activity on the RX line.

The Auto-Wake-up feature is enabled by setting both the WUE bit in the UxCON1 register and the UxIE bit in the P1Ex register. Once set, the normal receive sequence on RX is disabled, and the UART remains in an Idle state, monitoring for a wake-up event independent of the CPU mode. A wake-up event consists of a transition out of the Idle state on the RX line. (This coincides with the start of a Break or a wake-up signal character for the LIN protocol.)

The UART module generates a WUIF interrupt coincident with the wake-up event. The interrupt is generated synchronously to the Q clocks in normal CPU operating modes (Figure 31-13), and asynchronously, if the device is in Sleep mode (Figure 31-14). The interrupt condition is cleared by clearing the WUIF bit in the UxUIR register. To generate an interrupt on a wake-up event, all the following bits must be set:

- UxIE bit in the P1Ex register
- PIE and GIE bits in the INTCON register

The WUE bit is automatically cleared by the transition to the Idle state on the RX line at the end of the Break. This signals to the user that the Break event is over. At this point, the UART module is in Idle mode, waiting to receive the next character.

## 31.17.3.1 Special Considerations

### **Break Character**

To avoid character errors or character fragments during a wake-up event, the wake-up character must be all zeros.

When the wake-up is enabled, the function works independent of the low time on the data stream. If the WUE bit is set and a valid non-zero character is received, the low time from the Start bit to the first rising edge will be interpreted as the wake-up event. The remaining bits of the character will be received as a fragmented character and subsequent characters can result in framing or overrun errors.

Therefore, the initial character of the transmission must be all zeros. This must be eleven or more bit times, 13-bit times recommended for LIN bus, or any number of bit times for standard RS-232 devices.

### **Oscillator Start-up Time**

Oscillator start-up time must be considered, especially in applications using oscillators with longer start-up intervals (i.e., LP, XT or HS/PLL modes). The Sync Break (or wake-up signal) character must be of sufficient length, and be followed by a sufficient interval, to allow enough time for the selected oscillator to start and provide proper initialization of the UART.

### **WUE Bit**

To ensure that no actual data is lost, check the RXIDL bit to verify that a receive operation is not in process before setting the WUE bit. If a receive operation is not occurring, the WUE bit may then be set just prior to entering the Sleep mode.

**FIGURE 31-13: AUTO-WAKE-UP BIT (WUE) TIMING DURING NORMAL OPERATION**

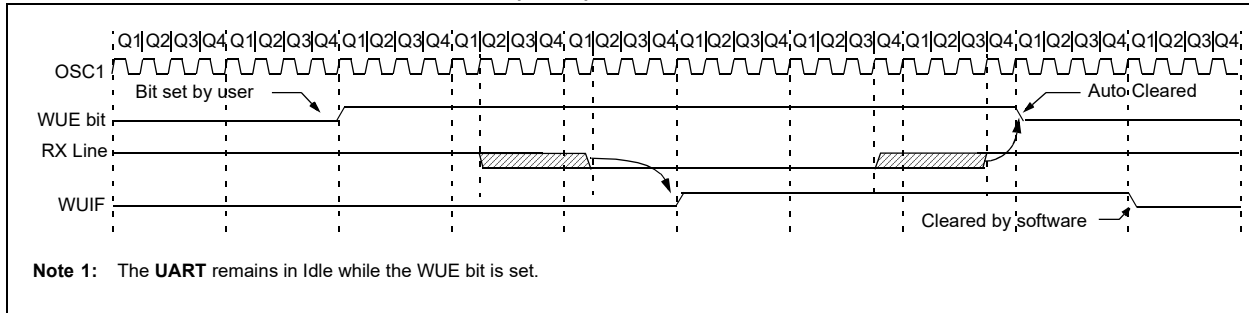
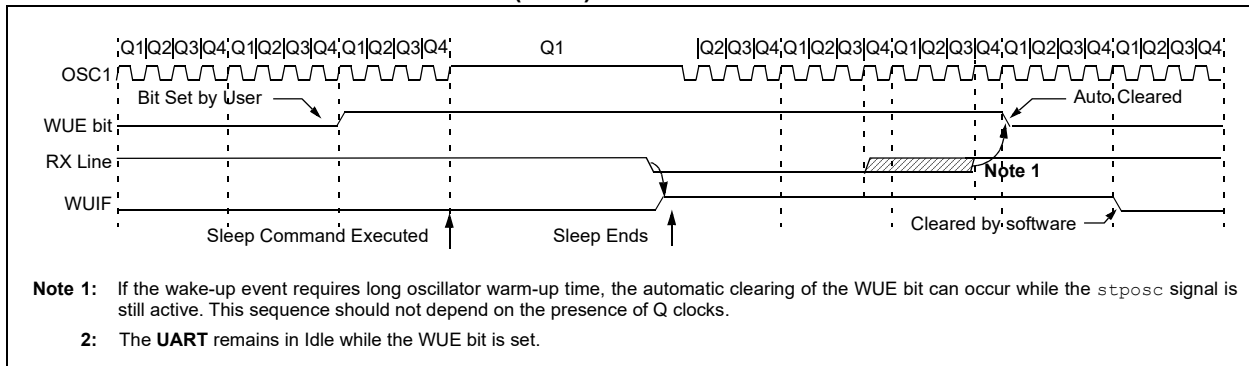


**FIGURE 31-14: AUTO-WAKE-UP BIT (WUE) TIMINGS DURING SLEEP**



## 31.18 Transmitting a Break

The UART module has the capability of sending either a fixed length Break period or a software timed Break period. The fixed length Break consists of a Start bit, followed by 12 '0' bits and a Stop bit. The software timed Break is generated by setting and clearing the BRKOVF bit in the UxCON1 register.

To send the fixed length Break, set the SENDB and TXEN bits in the UxCON0 register. The Break sequence is then initiated by a write to UxTXB. The timed Break will occur first, followed by the character written to UxTXB that initiated the Break. The initiating character is typically the Sync character of the LIN specification.

SENB is disabled in the LIN and DMX modes because those modes generate the Break sequence automatically.

The SENDB bit is automatically reset by hardware after the Break Stop bit is complete.

The TXMTIF bit in the UxERRIR register indicates when the transmit operation is active or idle, just as it does during normal transmission. See [Figure 31-15](#) for the timing of the Break sequence.

## 31.19 Receiving a Break

The UART has counters to detect when the RX input remains in the space state for an extended period of time. When this happens, the RXBKIF bit in the UxERRIR register is set.

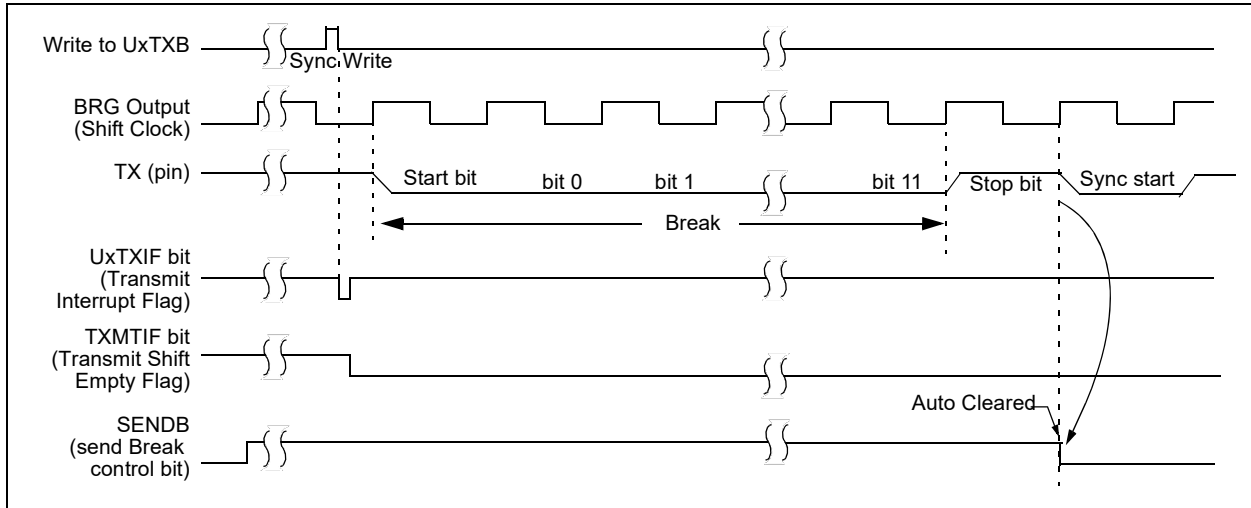
A Break is detected when the RX input remains in the space state for 11 bit periods for Asynchronous and LIN modes, and 23 bit periods for DMX mode.

The user can select to receive the Break interrupt as soon as the Break is detected or at the end of the Break, when the RX input returns to the Idle state. When the RXBIMD bit in the UxCON1 is '1' then RXBKIF is set immediately upon Break detection. When RXBIMD is '0' then RXBKIF is set when the RX input returns to the Idle state.

## 31.20 UART Operation During Sleep

The UART ceases to operate during Sleep. The safe way to wake the device from Sleep by a serial operation is to use the Wake-on-Break feature of the UART. See [Section 31.17.3, Auto-Wake-up on Break](#)

**FIGURE 31-15: SEND BREAK CHARACTER SEQUENCE**



## 31.21 Register Definitions: UART Control

Long bit name prefixes for the UART peripherals are shown below. Refer to [Section 1.3 “Register and Bit naming conventions”](#) for more information.

| Peripheral | Bit Name Prefix |
|------------|-----------------|
| UART 1     | U1              |
| UART 2     | U2              |

### REGISTER 31-1: UxCON0: UART CONTROL REGISTER 0

| R/W-0/0 | R/W/HS/HC-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0   | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|---------------|---------|---------|-----------|---------|---------|---------|
| BRGS    | ABDEN         | TXEN    | RXEN    | MODE<3:0> |         |         |         |
| bit 7   |               |         |         | bit 0     |         |         |         |

#### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared | HC = Hardware clear                                   |

|         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| bit 7   | <b>BRGS:</b> Baud rate Generator Speed Select bit<br>1 = Baud rate generator is high speed with 4 baud clocks per bit<br>0 = Baud rate generator is normal speed with 16 baud clocks per bit                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| bit 6   | <b>ABDEN:</b> Auto-baud Detect Enable bit <sup>(3)</sup><br>1 = Auto-baud is enabled. Receiver is waiting for Sync character (0x55)<br>0 = Auto-baud is not enabled or auto-baud is complete                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| bit 5   | <b>TXEN:</b> Transmit Enable Control bit <sup>(2)</sup><br>1 = Transmit is enabled. TX output pin drive is forced on when transmission is active, and controlled by PORT TRIS control when transmission is idle.<br>0 = Transmit is disabled. TX output pin drive is controlled by PORT TRIS control                                                                                                                                                                                                                                                                                                                            |
| bit 4   | <b>RXEN:</b> Receive Enable Control bit <sup>(2)</sup><br>1 = Receiver is enabled<br>0 = Receiver is disabled                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| bit 3-0 | <b>MODE&lt;3:0&gt;:</b> UART Mode Select bits <sup>(1)</sup><br>1111 = Reserved<br>1110 = Reserved<br>1101 = Reserved<br>1100 = LIN Master/Slave mode<br>1011 = LIN Slave-Only mode<br>1010 = DMX mode<br>1001 = DALI Control Gear mode<br>1000 = DALI Control Device mode<br>0111 = Reserved<br>0110 = Reserved<br>0101 = Reserved<br>0100 = Asynchronous 9-bit UART Address mode. 9th bit: 1 = address, 0 = data<br>0011 = Asynchronous 8-bit UART mode with 9th bit even parity<br>0010 = Asynchronous 8-bit UART mode with 9th bit odd parity<br>0001 = Asynchronous 7-bit UART mode<br>0000 = Asynchronous 8-bit UART mode |

- Note**
- 1: Changing the UART MODE while ON = 1 may cause unexpected results.
  - 2: Clearing TXEN or RXEN will not clear the corresponding buffers. Use TXBE or RXBE to clear the buffers.
  - 3: ABDEN is read-only when MODE = 1001. When MODE = 100x and ABDEN = 1, then auto-baud is determined from Start bit.

## REGISTER 31-2: UxCON1: UART CONTROL REGISTER 1

| R/W-0/0 | U-0 | U-0 | R/W/HC-0/0 | R/W-0/0 | U-0 | R/W-0/0 | R/W/HC-0/0 |
|---------|-----|-----|------------|---------|-----|---------|------------|
| ON      | —   | —   | WUE        | RXBIMD  | —   | BRKOVr  | SENDB      |
| bit 7   |     |     |            |         |     | bit 0   |            |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

HC = Hardware clear

bit 7 **ON:** Serial Port Enable bit

1 = Serial port enabled

0 = Serial port disabled (held in Reset)

bit 6-5 **Unimplemented:** Read as '0'

bit 4 **WUE:** Wake-up Enable bit

1 = Receiver is waiting for falling RX input edge which will set the UxIF bit. Cleared by hardware on wake event. Also requires UxIE bit of PEx to enable wake

0 = Receiver operates normally

bit 3 **RXBIMD:** Receive Break Interrupt Mode Select bit

1 = Set RXBKIF immediately when RX in has been low for the minimum Break time

0 = Set RXBKIF on rising RX input after RX in has been low for the minimum Break time

bit 2 **Unimplemented:** Read as '0'

bit 1 **BRKOVr:** Send Break Software Override bit

1 = TX output is forced to non-idle state

0 = TX output is driven by transmit shift register

bit 0 **SENDB:** Send Break Control bit<sup>(1)</sup>

1 = Output Break upon UxTXB write. Written byte follows Break. Bit is cleared by hardware.

0 = Break transmission completed or disabled

**Note 1:** This bit is read-only in LIN, DMX, and DALI modes.

## REGISTER 31-3: UxCON2: UART CONTROL REGISTER 2

| R/W-0/0 | R/W-0/0 | R/W-0/0  | R/W-0/0 | R/W-0/0 | R/W-0/0  | R/W-0/0 | R/W-0/0 |
|---------|---------|----------|---------|---------|----------|---------|---------|
| RUNOVF  | RXPOL   | STP<1:0> | C0EN    | TXPOL   | FLO<1:0> |         |         |
| bit 7   |         |          |         |         |          |         | bit 0   |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

- bit 7 **RUNOVF**: Run During Overflow Control bit  
 1 = RX input shifter continues to synchronize with Start bits after overflow condition  
 0 = RX input shifter stops all activity on receiver overflow condition
- bit 6 **RXPOL**: Receive Polarity Control bit  
 1 = Invert RX polarity, Idle state is low  
 0 = RX polarity is not inverted, Idle state is high
- bit 5-4 **STP<1:0>**: Stop Bit Mode Control bits<sup>(1)</sup>  
 11 = Transmit 2 Stop bits, receiver verifies first Stop bit  
 10 = Transmit 2 Stop bits, receiver verifies first and second Stop bits  
 01 = Transmit 1.5 Stop bits, receiver verifies first Stop bit  
 00 = Transmit 1 Stop bit, receiver verifies first Stop bit
- bit 3 **C0EN**: Checksum Mode Select bit  
**LIN mode**:  
 1 = Checksum Mode 1, enhanced LIN checksum includes PID in sum  
 0 = Checksum Mode 0, legacy LIN checksum does not include PID in sum  
**Other modes**:  
 1 = Add all TX and RX characters  
 0 = Checksums disabled
- bit 2 **TXPOL**: Transmit Polarity Control bit  
 1 = Output data is inverted, TX output is low in Idle state  
 0 = Output data is not inverted, TX output is high in Idle state
- bit 1-0 **FLO<1:0>**: Handshake Flow Control bits  
 11 = Reserved  
 10 = RTS/CTS and TXDE Hardware flow control  
 01 = XON/XOFF Software flow control  
 00 = Flow control is off

**Note 1:** All modes transmit selected number of Stop bits. Only DMX and DALI receivers verify selected number of Stop bits and all others verify only the first Stop bit.

## REGISTER 31-4: UxERRIR: UART ERROR INTERRUPT FLAG REGISTER

| R/S/C-1/1 | R/S/C-0/0 | R/W/S-0/0 | R/W/S-0/0 | R/S/C-0/0 | R/W/S-0/0 | R/W/S-0/0 | R/W/S-0/0 |
|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| TXMTIF    | PERIF     | ABDOVF    | CERIF     | FERIF     | RXBKIF    | RXFOIF    | TXCIF     |
| bit 7     |           |           |           |           |           |           | bit 0     |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared | S = Hardware set      C = Hardware clear              |

- bit 7      **TXMTIF:** Transmit Shift Register Empty Interrupt Flag bit  
1 = Transmit shift register is empty (Set at end of Stop bits)  
0 = Transmit shift register is actively shifting data
- bit 6      **PERIF:** Parity Error Interrupt Flag bit  
LIN and Parity modes:  
1 = Unread byte at top of input FIFO has parity error  
0 = Unread byte at top of input FIFO does not have parity error  
DALI Device mode:  
1 = Unread byte at top of input FIFO received as Forward Frame  
0 = Unread byte at top of input FIFO received as Back Frame  
Address mode:  
1 = Unread byte at top of input FIFO received as address  
0 = Unread byte at top of input FIFO received as data  
Other modes:  
Not used
- bit 5      **ABDOVF:** Auto-Baud Detect Overflow Interrupt Flag bit  
DALI mode:  
1 = Start bit measurement overflowed counter  
0 = No overflow during Start bit measurement  
Other modes:  
1 = Baud rate generator overflowed during the auto detection sequence  
0 = Baud rate generator has not overflowed
- bit 4      **CERIF:** Checksum Error Interrupt Flag bit (LIN mode only)  
1 = Checksum error  
0 = No checksum error
- bit 3      **FERIF:** Framing Error Interrupt Flag bit  
1 = Unread byte at top of input FIFO has framing error  
0 = Unread byte at top of input FIFO does not have framing error
- bit 2      **RXBKIF:** Break Reception Interrupt Flag bit  
1 = Break detected  
0 = No Break detected
- bit 1      **RXFOIF:** Receive FIFO Overflow Interrupt Flag bit  
1 = Receive FIFO has overflowed  
0 = Receive FIFO has not overflowed
- bit 0      **TXCIF:** Transmit Collision Interrupt Flag bit  
1 = Transmitted word is not equal to the word received during transmission  
0 = Transmitted word equals the word received during transmission

## REGISTER 31-5: UxERRIE: UART ERROR INTERRUPT ENABLE REGISTER

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| TXMTIE  | PERIE   | ABDOVE  | CERIE   | FERIE   | RXBKIE  | RXFOIE  | TXCIE   |
| bit 7   |         |         |         |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

- bit 7 **TXMTIE:** Transmit Shift Register Empty Interrupt Enable bit
  - 1 = Interrupt enabled
  - 0 = Interrupt not enabled
- bit 6 **PERIE:** Parity Error Interrupt Enable bit
  - 1 = Interrupt enabled
  - 0 = Interrupt not enabled
- bit 5 **ABDOVE:** Auto-Baud Detect Overflow Interrupt Enable bit
  - 1 = Interrupt enabled
  - 0 = Interrupt not enabled
- bit 4 **CERIE:** Checksum Error Interrupt Enable bit
  - 1 = Interrupt enabled
  - 0 = Interrupt not enabled
- bit 3 **FERIE:** Framing Error Interrupt Enable bit
  - 1 = Interrupt enabled
  - 0 = Interrupt not enabled
- bit 2 **RXBKIE:** Break Reception Interrupt Enable bit
  - 1 = Interrupt enabled
  - 0 = Interrupt not enabled
- bit 1 **RXFOIE:** Receive FIFO Overflow Interrupt Enable bit
  - 1 = Interrupt enabled
  - 0 = Interrupt not enabled
- bit 0 **TXCIE:** Transmit Collision Interrupt Enable bit
  - 1 = Interrupt enabled
  - 0 = Interrupt not enabled



## REGISTER 31-6: UxUIR: UART GENERAL INTERRUPT REGISTER

| R/S/W-0/0 | R/S/W-0/0 | U-0 | U-0 | U-0 | R/W-0/0 | U-0 | U-0   |
|-----------|-----------|-----|-----|-----|---------|-----|-------|
| WUIF      | ABDIF     | —   | —   | —   | ABDIE   | —   | —     |
| bit 7     |           |     |     |     |         |     | bit 0 |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared | S = Hardware set                                      |

- bit 7      **WUIF:** Wake-up Interrupt bit  
1 = Idle to non-idle transition on RX line detected when WUE is set. Also sets UxIF. (WUIF must be cleared by software to clear UxIF)  
0 = WUE not enabled by software or no transition detected
- bit 6      **ABDIF:** Auto-Baud detect interrupt bit  
1 = Auto-baud detection complete. Status shown in UxIF when ABDIE is set. (Must be cleared by software)  
0 = Auto-baud not enabled or auto-baud enabled and auto-baud detection not complete
- bit 5-3      **Unimplemented:** Read as '0'
- bit 2      **ABDIE:** Auto-Baud Detect Interrupt Enable bit  
1 = ABDIF will set UxIF bit in PIRx register  
0 = ABDIF will not set UxIF
- bit 1-0      **Unimplemented:** Read as '0'

## REGISTER 31-7: Ux FIFO: UART FIFO STATUS REGISTER

| R/W/S-0/0 | R/W-0/0 | R/W/S/C-1/1 | R/S/C-0/0 | R/S/C-1/1 | S/C-1/1 | R/W/S/C-1/1 | R/S/C-0/0 |
|-----------|---------|-------------|-----------|-----------|---------|-------------|-----------|
| TXWRE     | STPMD   | TXBE        | TXBF      | RXIDL     | XON     | RXBE        | RXBF      |
| bit 7     |         |             |           |           |         |             | bit 0     |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared | S = Hardware set      C = Hardware clear              |

- bit 7 **TXWRE:** Transmit Write Error Status bit (Must be cleared by software)
- LIN Master mode:  
1 = UxP1L was written when a master process was active
- LIN Slave mode:  
1 = UxTXB was written when UxP2 = 0 or more than UxP2 bytes have been written to UxTXB since last Break
- Address Detect mode:  
1 = UxP1L was written before the previous data in UxP1L was transferred to TX shifter
- All modes:  
1 = A new byte was written to UxTXB when the output FIFO was full  
0 = No error
- bit 6 **STPMD:** Stop Bit Detection Mode bit  
1 = Assert UxRXIF at end of last Stop bit or end of first Stop bit when STP = 11  
0 = Assert UxRXIF in middle of first Stop bit
- bit 5 **TXBE:** Transmit Buffer Empty Status bit  
1 = Transmit buffer is empty. Setting this bit will clear the transmit buffer and output shift register.  
0 = Transmit buffer is not empty. Software cannot clear this bit.
- bit 4 **TXBF:** Transmit Buffer Full Status bit  
1 = Transmit buffer is full  
0 = Transmit buffer is not full
- bit 3 **RXIDL:** Receive Pin Idle Status bit  
1 = Receive pin is in Idle state  
0 = UART is receiving Start, Stop, Data, Auto-baud, or Break
- bit 2 **XON:** Software Flow Control Transmit Enable Status bit  
1 = Transmitter is enabled  
0 = Transmitter is disabled
- bit 1 **RXBE:** Receive Buffer Empty Status bit  
1 = Receive buffer is empty. Setting this bit will clear the RX buffer<sup>(1)</sup>  
0 = Receive buffer is not empty. Software cannot clear this bit.
- bit 0 **RXBF:** Receive Buffer Full Status bit  
1 = Receive buffer is full  
0 = Receive buffer is not full

**Note 1:** The BSF instruction should not be used to set RXBE because doing so will clear a byte pending in the transmit shift register when the UxTXB register is empty. Instead, use the MOVWF instruction with a '0' in the TXBE bit location.

## REGISTER 31-8: UxBRGL: UART BAUD RATE GENERATOR LOW REGISTER

|          |         |         |         |         |         |         |         |
|----------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0  | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| BRG<7:0> |         |         |         |         |         |         |         |
| bit 7    |         |         |         | bit 0   |         |         |         |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0 **BRG<7:0>**: Least Significant Byte of Baud Rate Generator

## REGISTER 31-9: UxBRGH: UART BAUD RATE GENERATOR HIGH REGISTER

|           |         |         |         |         |         |         |         |
|-----------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0   | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| BRG<15:8> |         |         |         |         |         |         |         |
| bit 7     |         |         |         | bit 0   |         |         |         |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0 **BRG<15:8>**: Most Significant Byte of Baud Rate Generator

- Note 1:** The UxBRG registers should only be written when ON = 0.
- 2:** Maximum BRG value when MODE = '100x' and BRGS = 1 is 0x7FFE.
- 3:** Maximum BRG value when MODE = '100x' and BRGS = 0 is 0x1FFE.

## REGISTER 31-10: UxRXB: UART RECEIVE REGISTER

|          |       |       |       |       |       |       |       |
|----------|-------|-------|-------|-------|-------|-------|-------|
| R-0/0    | R-0/0 | R-0/0 | R-0/0 | R-0/0 | R-0/0 | R-0/0 | R-0/0 |
| RXB<7:0> |       |       |       |       |       |       |       |
| bit 7    |       |       |       | bit 0 |       |       |       |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-0      **RXB<7:0>**: Top of Receive Buffer

## REGISTER 31-11: UxTXB: UART TRANSMIT REGISTER

|          |         |         |         |         |         |         |         |
|----------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0  | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| TXB<7:0> |         |         |         |         |         |         |         |
| bit 7    |         |         |         | bit 0   |         |         |         |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-0      **TXB<7:0>**: Bottom of Transmit Buffer

## REGISTER 31-12: UxP1H: UART PARAMETER 1 HIGH REGISTER

|       |     |     |     |     |     |     |         |
|-------|-----|-----|-----|-----|-----|-----|---------|
| U-0   | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0/0 |
| —     | —   | —   | —   | —   | —   | —   | P1<8>   |
| bit 7 |     |     |     |     |     |     | bit 0   |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-6 **Unimplemented:** Read as '0'

bit 0 **P1<8>:** Most Significant Bit of Parameter 1

#### DMX mode:

Most Significant bit of number of bytes to transmit between Start Code and automatic Break generation

#### DALI Control Device mode:

Most Significant bit of idle time delay after which a Forward Frame is sent. Measured in half-bit periods

#### DALI Control Gear mode:

Most Significant bit of delay between the end of a Forward Frame and the start of the Back Frame  
Measured in half-bit periods

#### Other modes:

Not used

## REGISTER 31-13: UxP1L: UART PARAMETER 1 LOW REGISTER

|         |         |         |         |         |         |         |         |
|---------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| P1<7:0> |         |         |         |         |         |         |         |
| bit 7   |         |         |         |         |         |         | bit 0   |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0 **P1<7:0>:** Least Significant Bits of Parameter 1

#### DMX mode:

Least Significant Byte of number of bytes to transmit between Start Code and automatic Break generation

#### DALI Control Device mode:

Least Significant Byte of idle time delay after which a Forward Frame is sent. Measured in half-bit periods

#### DALI Control Gear mode:

Least Significant Byte of delay between the end of a Forward Frame and the start of the Back Frame  
Measured in half-bit periods

#### LIN mode:

PID to transmit (Only Least Significant 6 bits used)

#### Asynchronous Address mode:

Address to transmit (9th transmit bit automatically set to '1')

#### Other modes:

Not used

## REGISTER 31-14: UxP2H: UART PARAMETER 2 HIGH REGISTER

|       |     |     |     |     |     |     |         |
|-------|-----|-----|-----|-----|-----|-----|---------|
| U-0   | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0/0 |
| —     | —   | —   | —   | —   | —   | —   | P2<8>   |
| bit 7 |     |     |     |     |     |     | bit 0   |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-6 **Unimplemented:** Read as '0'

bit 0 **P2<8>:** Most Significant Bit of Parameter 2

#### DMX mode:

Most Significant bit of first address of receive block

#### DALI mode:

Most Significant bit of number of half-bit periods of idle time in Forward Frame detection threshold

#### Other modes:

Not used

## REGISTER 31-15: UxP2L: UART PARAMETER 2 LOW REGISTER

|         |         |         |         |         |         |         |         |
|---------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| P2<7:0> |         |         |         |         |         |         |         |
| bit 7   |         |         |         |         |         |         | bit 0   |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0 **P2<7:0>:** Least Significant Bits of Parameter 2

#### DMX mode:

Least Significant Byte of first address of receive block

#### LIN Slave mode:

Number of data bytes to transmit

#### DALI mode:

Least Significant Byte of number of half-bit periods of idle time in Forward Frame detection threshold

#### Asynchronous Address mode:

Receiver address

#### Other modes:

Not used

## REGISTER 31-16: UxP3H: UART PARAMETER 3 HIGH REGISTER

|       |     |     |     |     |     |     |         |
|-------|-----|-----|-----|-----|-----|-----|---------|
| U-0   | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0/0 |
| —     | —   | —   | —   | —   | —   | —   | P3<8>   |
| bit 7 |     |     |     |     |     |     | bit 0   |

### Legend:

R = Readable bit      W = Writable bit      U = Unimplemented bit, read as '0'  
 u = Bit is unchanged      x = Bit is unknown      -n/n = Value at POR and BOR/Value at all other Resets  
 '1' = Bit is set      '0' = Bit is cleared

bit 7-6      **Unimplemented:** Read as '0'  
 bit 0      **P3<8>:** Most Significant Bit of Parameter 3  
             DMX mode:  
             Most Significant bit of last address of receive block  
             Other modes:  
             Not used

## REGISTER 31-17: UxP3L: UART PARAMETER 3 LOW REGISTER

|         |         |         |         |         |         |         |         |
|---------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| P3<7:0> |         |         |         |         |         |         |         |
| bit 7   |         |         |         |         |         |         | bit 0   |

### Legend:

R = Readable bit      W = Writable bit      U = Unimplemented bit, read as '0'  
 u = Bit is unchanged      x = Bit is unknown      -n/n = Value at POR and BOR/Value at all other Resets  
 '1' = Bit is set      '0' = Bit is cleared

bit 7-0      **P3<7:0>:** Least Significant Bits of Parameter 3  
             DMX mode:  
             Least Significant Byte of last address of receive block  
             LIN Slave mode:  
             Number of data bytes to receive  
             Asynchronous Address mode:  
             Receiver address mask. Received address is XOR'd with UxP2L then AND'd with UxP3L  
             Match occurs when result is zero  
             Other modes:  
             Not used

## REGISTER 31-18: UxTXCHK: UART TRANSMIT CHECKSUM RESULT REGISTER

|            |         |         |         |         |         |         |         |
|------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0    | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| TXCHK<7:0> |         |         |         |         |         |         |         |
| bit 7      |         |         |         |         |         |         | bit 0   |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0 **TXCHK<7:0>**: Checksum calculated from TX bytes

LIN mode and C0EN = 1:

Sum of all transmitted bytes including PID

LIN mode and C0EN = 0:

Sum of all transmitted bytes except PID

All other modes and C0EN = 1:

Sum of all transmitted bytes since last clear

All other modes and C0EN = 0:

Not used

## REGISTER 31-19: UxRXCHK: UART RECEIVE CHECKSUM RESULT REGISTER

|            |         |         |         |         |         |         |         |
|------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0    | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| RXCHK<7:0> |         |         |         |         |         |         |         |
| bit 7      |         |         |         |         |         |         | bit 0   |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0 **RXCHK<7:0>**: Checksum calculated from RX bytes

LIN mode and C0EN = 1:

Sum of all received bytes including PID

LIN mode and C0EN = 0:

Sum of all received bytes except PID

All other modes and C0EN = 1:

Sum of all received bytes since last clear

All other modes and C0EN = 0:

Not used



**TABLE 31-3: SUMMARY OF REGISTERS ASSOCIATED WITH THE UART**

| Name    | Bit 7      | Bit 6 | Bit 5    | Bit 4 | Bit 3     | Bit 2  | Bit 1    | Bit 0 | Register on page    |
|---------|------------|-------|----------|-------|-----------|--------|----------|-------|---------------------|
| UxCON0  | BRGS       | ABDEN | TXEN     | RXEN  | MODE<3:0> |        |          |       | <a href="#">484</a> |
| UxCON1  | ON         | —     | —        | WUE   | RXBIMD    | —      | BRKOVF   | SENDB | <a href="#">485</a> |
| UxCON2  | RUNOVF     | RXPOL | STP<1:0> |       | C0EN      | TXPOL  | FLO<1:0> |       | <a href="#">486</a> |
| UxERRIR | TXMTIF     | PERIF | ABDOVF   | CERIF | FERIF     | RXBKIF | RXFOIF   | TXCIF | <a href="#">487</a> |
| UxERRIE | TXMTIE     | PERIE | ABDOVE   | CERIE | FERIE;    | RXBKIE | RXFOIE   | TXCIE | <a href="#">488</a> |
| UxUIR   | WUIF       | ABDIF | —        | —     | —         | ABDIE  | —        | —     | <a href="#">489</a> |
| UxFIFO  | TXWRE      | STPMD | TXBE     | TXBF  | RXIDL     | XON    | RXBE     | RXBF  | <a href="#">490</a> |
| UxBRGL  | BRG<7:0>   |       |          |       |           |        |          |       | <a href="#">491</a> |
| UxBRGH  | BRG<15:8>  |       |          |       |           |        |          |       | <a href="#">491</a> |
| UxRXB   | RXB<7:0>   |       |          |       |           |        |          |       | <a href="#">492</a> |
| UxTXB   | TXB<7:0>   |       |          |       |           |        |          |       | <a href="#">492</a> |
| UxP1H   | —          | —     | —        | —     | —         | —      | —        | P1<8> | <a href="#">493</a> |
| UxP1L   | P1<7:0>    |       |          |       |           |        |          |       | <a href="#">493</a> |
| UxP2H   | —          | —     | —        | —     | —         | —      | —        | P2<8> | <a href="#">494</a> |
| UxP2L   | P2<7:0>    |       |          |       |           |        |          |       | <a href="#">494</a> |
| UxP3H   | —          | —     | —        | —     | —         | —      | —        | P3<8> | <a href="#">495</a> |
| UxP3L   | P3<7:0>    |       |          |       |           |        |          |       | <a href="#">495</a> |
| UxTXCHK | TXCHK<7:0> |       |          |       |           |        |          |       | <a href="#">496</a> |
| UxRXCHK | RXCHK<7:0> |       |          |       |           |        |          |       | <a href="#">496</a> |

**Legend:** — = unimplemented, read as '0'. Shaded cells are unused by the UART module.

## 32.0 SERIAL PERIPHERAL INTERFACE (SPI) MODULE

### 32.1 SPI Module Overview

The SPI (Serial Peripheral Interface) module is a synchronous serial data communication bus that operates in Full-Duplex mode. Devices communicate in a master/slave environment where the master device initiates the communication. A slave device is controlled through a Chip Select known as Slave Select. Example slave devices include serial EEPROMs, shift registers, display drivers, A/D converters, or another PIC® device.

The SPI bus specifies four signal connections:

- Serial Clock (SCK)
- Serial Data Out (SDO)
- Serial Data IN (SDI)
- Slave Select ( $\overline{SS}$ )

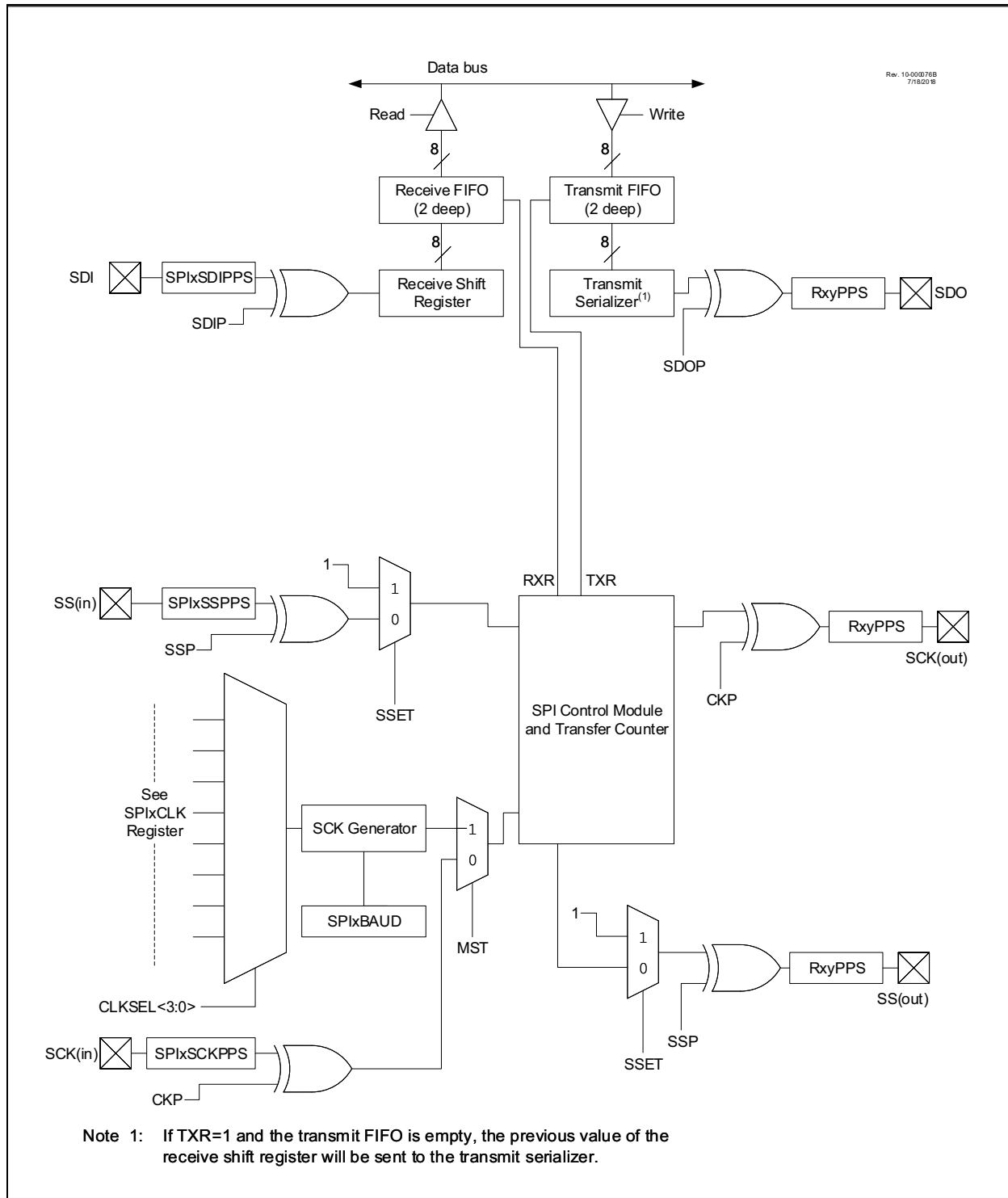
The SPI interface supports the following modes and features:

- Master mode
- Slave mode
- Clock Polarity and Edge Select
- SDI, SDO, and SS Polarity Control
- Separate Transmit and Receive Enables
- Slave Select Synchronization
- Daisy-chain connection of slave devices
- Separate Transmit and Receive Buffers with 2-byte FIFO and DMA capabilities

Figure 32-1 shows the block diagram of the SPI module.

# PIC18(L)F25/26K83

**FIGURE 32-1: SPI MODULE SIMPLIFIED BLOCK DIAGRAM**



The SPI transmit output (SDO\_out) is available to the remappable PPS SDO pin and internally to the following peripherals:

- Configurable Logic Cell (CLC)
- Data Signal Modulator (DSM)

The SPI bus typically operates with a single master device and one or more slave devices. When multiple slave devices are used, an independent Slave Select connection is required from the master device to each slave device.

The master selects only one slave at a time. Most slave devices have tri-state outputs so their output signal appears disconnected from the bus when they are not selected.

Transmissions typically involve shift registers, eight bits in size, one in the master and one in the slave. With either the master or the slave device, data is always shifted out one bit at a time, with the Most Significant bit (MSb) shifted out first. At the same time, a new bit is shifted into the device. Unlike older Microchip devices, the SPI on the PIC18(L)F25/26K83 contains two separate registers for incoming and outgoing data. Both registers also have 2-byte FIFO buffers and allow for DMA bus connections.

Figure 32-2 shows a typical connection between two devices configured as master and slave devices.

Data is shifted out of the transmit FIFO on the programmed clock edge and into the receive shift register on the opposite edge of the clock.

The master device transmits information on its SDO output pin which is connected to, and received by, the slave's SDI input pin. The slave device transmits information on its SDO output pin, which is connected to, and received by, the master's SDI input pin.

The master device sends out the clock signal. Both the master and the slave devices should be configured for the same clock polarity.

During each SPI clock cycle, a full-duplex data transmission occurs. This means that while the master device is sending out the MSb from its output register (on its SDO pin) and the slave device is reading this bit and saving as the LSb of its input register, that the slave device is also sending out the MSb from its shift register (on its SDO pin) and the master device is reading this bit and saving it as the LSb of its input register.

After eight bits have been shifted out, the master and slave have exchanged register values and stored the incoming data into the receiver FIFOs.

If there is more data to exchange, the registers are loaded with new data and the process repeats itself.

Whether the data is meaningful or not (dummy data) depends on the application software. This leads to three scenarios for data transmission:

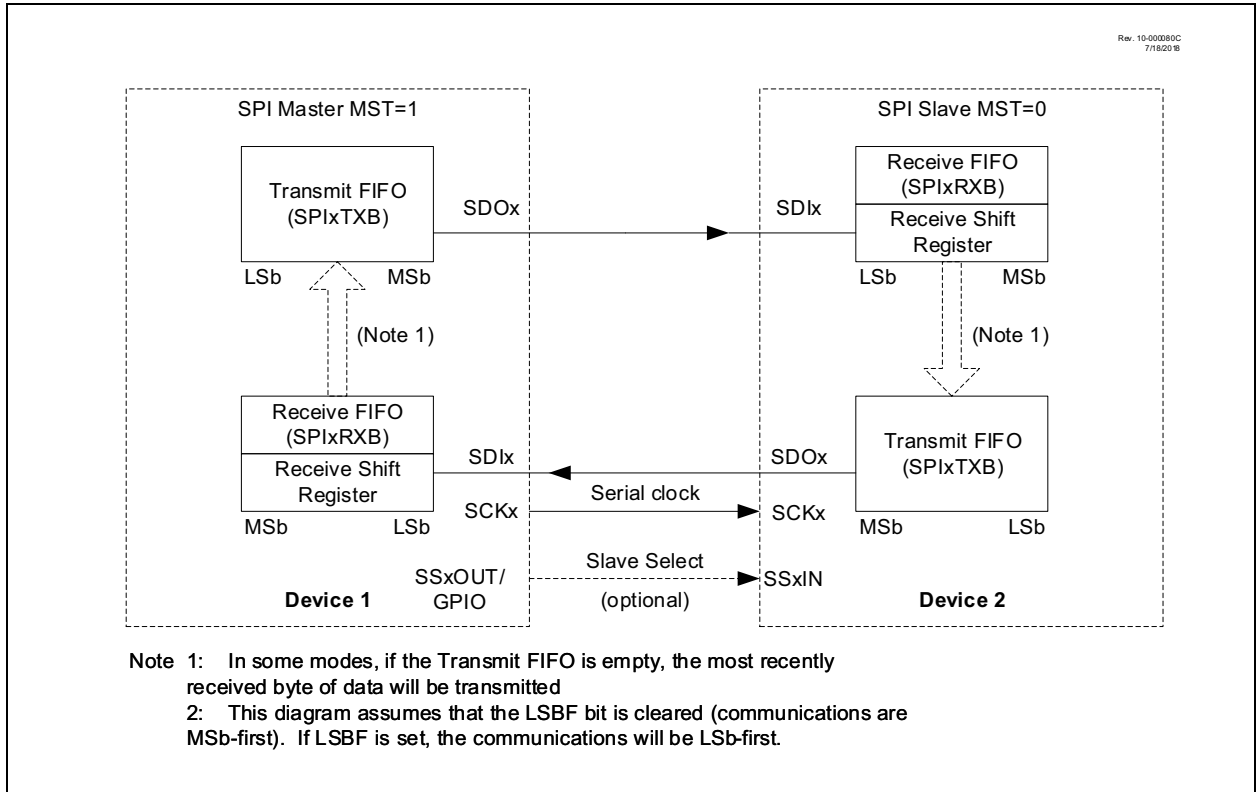
- Master sends useful data and slave sends dummy data
- Master sends useful data and slave sends useful data
- Master sends dummy data and slave sends useful data

In this particular SPI module, dummy data may be sent without software involvement, by clearing either the RXR bit (for receiving dummy data) or the TXR bit (for sending dummy data) (see Table 32-1 as well as Section 32.5 “Master mode” and Section 32.6 “Slave Mode” for further TXR/RXR setting details). This SPI module can send transmissions of any number of bits, and can send information in segments of varying size (from 1-8 bits in width). As such, transmissions may involve any number of clock cycles, depending on the amount of data to be transmitted.

When there is no more data to be transmitted, the master stops sending the clock signal and deselects the slave.

Every slave device connected to the bus that has not been selected through its Slave Select line disregards the clock and transmission signals and does not transmit out any data of its own.

**FIGURE 32-2: SPI MASTER/SLAVE CONNECTION WITH FIFOs**



## 32.2 SPI REGISTERS

- SPI Interrupt Flag Register (SPIxINTF)
- SPI Interrupt Enable Register (SPIxINTE)
- SPI Byte Count High and Low Registers (SPIxTCTH/L)
- SPI Bit Count Register (SPIxTWIDTH)
- SPI Baud Rate Register (SPIxBAUD)
- SPI Control Register 0 (SPIxCON0)
- SPI Control Register 1 (SPIxCON1)
- SPI Control Register 2 (SPIxCON2)
- SPI FIFO Status Register (SPIxSTATUS)
- SPI Receiver Buffer Register (SPIxRB)
- SPI Transmit Buffer Register (SPIxTB)
- SPI Clock Select Register (SPIxCLK)

SPIxCON0, SPIxCON1, and SPIxCON2 are control registers for the SPI module.

SPIxSTATUS contains several Status bits that indicate the status of both the SPI module and the receive and transmit FIFOs.

SPIxBAUD and SPIxCLK control the Baud Rate Generator (BRG) of the SPI module when in Master mode. The SPIxCLK selects the clock source that is used. The SPIxBAUD configures the clock divider used on that clock. More information on the baud rate generator is available in [Section 32.5.6 “Master Mode SPI Clock Configuration”](#).

SPIxTxB and SPIxRxB are the transmit and receive buffer registers used to send and receive data on the SPI bus. They both offer indirect access to shift registers that are used for shifting the data in and out. Both registers access the two-byte FIFOs, allowing for multiple transmissions/receptions to be stored between software transfers the data.

The SPIxTCTH:L register pair either count or control the number of bits or bytes in a data transfer. When BMODE = 1, the SPIxTCT value signifies bytes and the SPIxTWIDTH value signifies the number of bits in a byte. When BMODE = 0, the SPIxTCT value is concatenated with the SPIxTWIDTH register to signify bits. In Master Receive-only mode (TXR = 0 and RXR = 1), the data transfer is initiated by writing SPIxTCT with the desired bit or byte value to transfer. In Master Transmit mode (TXR = 1), the data transfer is initiated by writing the SPIxTxB register, in which case the SPIxTCT is a down counter for the bits or bytes transferred.

The SPIxINTF and SPIxINTE are the flags and enables, respectively, for SPI-specific interrupts. They are tied to the SPIxIF flag and SPIxIE enable in the PIR and PIE registers, which is triggered when any interrupt contained in the SPIxINTF/SPIxINTE registers is triggered. The PIR/PIE registers also contain SPIxTXIF/SPIxTXIE bits, which are the interrupt flag and enable for the SPI Transmit Interrupt, as well as the SPIxRXIF/SPIxRXIE bits, which are the interrupt flag and enable for the SPI Receive Interrupt.

## 32.3 SPI MODE OPERATION

When initializing the SPI, several options need to be specified. This is done by programming the appropriate control bits (SPIxCON0<2:0>, SPIxCON1<7:4>, SPIxCON1<2:0>, and SPIxCON2<2:0>). These control bits allow the following to be specified:

- Master mode (SCK is the clock output)
- Slave mode (SCK is the clock input)
- Clock Polarity (Idle state of SCK)
- Input, Output, and Slave Select Polarity
- Data Input Sample Phase (middle or end of data output time)
- Clock Edge (output data on first/second edge of SCK)
- Clock Rate (Master mode only)
- Slave Select Mode (Master or Slave mode)
- MSB-First or LSB-First
- Receive/Transmit Modes
  - Full-duplex
  - Receive-without-transmit
  - Transmit-without-receive
- Transfer Counter Mode (Transmit-without-receive mode)

## 32.3.1 ENABLING AND DISABLING THE SPI MODULE

To enable the serial peripheral, the SPI enable bit (EN in SPIxCON0) must be set. To reset or reconfigure SPI mode, clear the EN bit, re-initialize the SSPxCONx registers and then set the EN bit. Setting the EN bit enables the SPI inputs and outputs: SDI, SDO, SCK(out), SCK(in), SS(out), and SS(in). All of these inputs and outputs are steered by PPS, and thus must have their functions properly mapped to device pins to function (see [Section 17.0 “Peripheral Pin Select \(PPS\) Module”](#)). In addition, SS(out) and SCK(out) must have the pins they are steered to set as outputs (TRIS bits must be ‘0’) in order to properly output. Clearing the TRIS bit of the SDO pin will cause the SPI module to always control that pin, but is not necessary for SDO functionality. (see [Section 32.3.5 “Input and Output Polarity Bits”](#)). Configurations selected by the following registers should not be changed while the EN bit is set:

- SPIxBAUD
- SPIxCON1
- SPIxCON0 (except to clear the EN bit)

Clearing the EN bit aborts any transmissions in progress, disables the setting of interrupt flags by hardware, and resets the FIFO occupancy (see [Section 32.3.3 “Transmit and Receive FIFOs”](#) for more FIFO details).

## 32.3.2 BUSY BIT

While a data transfer is in progress, the SPI module sets the BUSY bit of SPIxCON2. This bit can be polled by the user to determine the current status of the SPI module, and to know when a communication is complete. The following registers/bits should not be written by software while the BUSY bit is set:

- SPIxTCNTH/L
- SPIxTWIDTH
- SPIxCON2
- The CLRBF bit of SPIxSTATUS

**Note:** It is also not recommended to read SPIxTCNTH/L while the BUSY bit is set, as the value in the registers may not be a reliable indicator of the Transfer Counter. Use the Transfer Count Zero Interrupt Flag (the TCZIF bit of SPIxINTF) to accurately determine that the Transfer Counter has reached zero.

## 32.3.3 TRANSMIT AND RECEIVE FIFOs

The transmission and reception of data from the SPI module is handled by two FIFOs, one for reception and one for transmission (addressed by the SFRs SPIxRXB and SPIxTXB, respectively.). The TXFIFO is written by software and is read by the SPI module to shift the data onto the SDO pin. The RXFIFO is written by the SPI module as it shifts in the data from the SDI pin and is read by software. Setting the CLRBF bit of SPIxSTATUS resets the occupancy for both FIFOs, emptying both buffers. The FIFOs are also reset by disabling the SPI module.

**Note:** TXFIFO occupancy and RXFIFO occupancy simply refer to the number of bytes that are currently being stored in each FIFO. These values are used in this chapter to illustrate the function of these FIFOs and are not directly accessible through software.

The SPIxRXB register addresses the receive FIFO and is read-only. Reading from this register will read from the first FIFO location that was written to by hardware and decrease the RXFIFO occupancy. If the FIFO is empty, reading from this register will instead return a value of zero and set the RXRE (Receive Buffer Read Error) bit of the SPIxSTATUS register. The RXRE bit must then be cleared in software in order to properly reflect the status of the read error. When RXFIFO is full, the RXBF bit of the SPIxSTATUS register will be set. When the device receives data on the SDI pin, the receive FIFO may be written to by hardware and the occupancy increased, depending on the mode and receiver settings, as summarized in [Table 32-1](#).

The SPIxTXB register addresses the transmit FIFO and is write-only. Writing to the register will write to the first empty FIFO location and increase the occupancy. If the FIFO is full, writing to this register will not affect the data and will set the TXWE bit of the SPIxSTATUS register. When the TXFIFO is empty, the TXBE bit of SPIxSTATUS will be set. When a data transfer occurs, data may be read from the first FIFO location written to and the occupancy decreases, depending on mode and transmitter settings, as summarized in [Table 32-1](#) and [Section 32.6.1 “Slave Mode Transmit options”](#).

## 32.3.4 LSB VS. MSB-FIRST OPERATION

Typically, SPI communication is output Most-Significant bit first, but some devices/buses may not conform to this standard. In this case, the LSBF bit may be used to alter the order in which bits are shifted out during the data exchange. In both Master and Slave mode, the LSBF bit of SPIxCON0 controls if data is shifted MSb or LSb first. Clearing the bit (default) configures the data to transfer MSb first, which is traditional SPI operation, while setting the bit configures the data to transfer LSb first.

## 32.3.5 INPUT AND OUTPUT POLARITY BITS

SPIxCON1 has three bits that control the polarity of the SPI inputs and outputs. The SDIP bit controls the polarity of the SDI input, the SDOP bit controls the polarity of the SDO output, and the SSP bit controls the polarity of both the slave  $\overline{SS}$  input and the master SS output. For all three bits, when the bit is clear, the input or output is active-high, and when the bit is set, the input or output is active-low. When the EN bit of SPIxCON0 is cleared, SS(out) and SCK(out) both revert to the inactive state dictated by their polarity bits. The SDO output state when the EN bit of SPIxCON0 is cleared is determined by several factors.

- When the associated TRIS bit for the SDO pin is cleared, and the SPI goes Idle after a transmission, the SDO output will remain at the last bit level. The SDO pin will revert to the Idle state if EN is cleared.
- When the associated TRIS bit for the SDO pin is set, behavior varies in Slave and Master mode.
  - In Slave mode, the SDO pin tri-states when:
    - Slave Select is inactive,
    - the EN bit of SPIxCON0 is cleared, or when
    - the TXR bit of SPIxCON2 is cleared.
  - In Master mode, the SDO pin tri-states when TXR = 0. When TXR = 1 and the SPI goes Idle after a transmission, the SDO output will remain at the last bit level. The SDO pin will revert to the Idle state if EN is cleared.

## 32.4 Transfer Counter

In all Master modes, the transfer counter can be used to determine how many data transfers the SPI will send/receive. The transfer counter is comprised of the SPIxTCTH/L set of registers, and is also partially controlled by the SPIxTWIDTH register. The Transfer Counter has two primary modes, determined by the BMODE bit of the SPIxCON0 register. Each mode uses the SPIxTCTH/L and SPIxTWIDTH registers to determine the number and size of the transfers. In both modes, when the transfer counter reaches zero, the TCZIF interrupt flag is set.

**Note:** When BMODE=1 in all Master modes (and at all times in Slave modes), the Transfer Counter will still decrement as transfers occur and can be used to count the number of messages sent/received, as well as to control SS(out) and to trigger TCZIF. Also when BMODE = 1, the SPIxTWIDTH register can be used in Master and Slave modes to determine the size of messages sent and received by the SPI, even if the Transfer Counter is not being actively used to control the number of messages being sent/received by the SPI module.

### 32.4.1 TOTAL BIT COUNT MODE (BMODE = 0)

In this mode, SPIxTCTH/L and SPIxTWIDTH are concatenated to determine the total number of bits to be transferred. These bits will be loaded from/into the transmit/receive FIFOs in 8-bit increments and the transfer counter will be decremented by eight until the total number of remaining bits is less than eight. If there are any remaining bits (SPIxTWIDTH  $\neq$  0), the transmit FIFO will send out one final message with any extra bits greater than the remainder ignored. The SPIxTWIDTH is the remaining bit count but the value does not change as it does for the SPIxTCT value. Similarly, the receiver will load a final byte into the receiver FIFO, and pad the extra bits with zeros. The LSBF bit of SPIxCON0 determines whether the Most Significant or Least Significant bits of this final byte are ignored/padded. For example, when LSBF = 0 and the final transfer contains only two bits then if the last byte sent was 5Fh then the RXB of the receiver will contain 40h which are the two MSbits of the final byte padded with zeros in the LSbits.

In this mode, the SPI master will only transmit messages when the SPIxTCT value is greater than zero, regardless of TXR and RXR settings. In Master Transmit mode, the transfer starts with the data write to the SPIxTXB register or the count value written to the SPIxTCTL register, whichever occurs last. In Master Receive-only mode, the transfer clocks start when the SPIxTCTL value is written. Transfer clocks are suspended when the receive FIFO is full and resume as the FIFO is read.



## 32.4.2 VARIABLE TRANSFER SIZE MODE (BMODE = 1)

In this mode, SPIxTWIDTH specifies the width of every individual piece of the data transfer in bits. SPIxTCTH/SPIxTCTL specifies the number of transfers of this bit length. If SPIxTWIDTH = 0, each piece is a full byte of data. If SPIxTWIDTH  $\neq$  0, then only the specified number of bits from the transmit FIFO are shifted out, with the unused bits ignored. Received data is padded with zeros in the unused bit areas when transferred into the receive FIFO. The LSBF bit of SPIxCON0 determines whether the Most Significant or Least Significant bits of the transfers are ignored/padded. In this mode, the transfer counter being zero only stops messages from being sent/received when in "Receive only" mode.

**Note:** With BMODE = 1, it is possible for the transfer counter (SPIxTCTH/L) to decrement below zero, although when in "Receive Only" Master mode, transfer clocks will cease when the transfer counter reaches zero.

## 32.4.3 TRANSFER COUNTER IN SLAVE MODE

In Slave mode, the transfer counter will still decrement as data is shifted in and out of the SPI module, but it will not control data transfers. In addition, in Slave mode, the BMODE bit along with the transfer counter is used to determine when the device should look for Slave Select faults. If BMODE = 0, the SSFLT bit will be set if Slave Select transitions from its active to inactive state during bytes of data, as well as if it transitions before the last bit sent during the final byte (if SPIxTWIDTH $\neq$ 0). If BMODE=1, the SSFLT bit will be set if Slave Select transitions from its active to inactive state before the final bit of each individual transfer is completed. Note that SSFLT does not have an associated interrupt, so it should be checked in software. An ideal time to do this is when the End of Slave Select Interrupt (EOSIF) is triggered (see [Section 32.8.3.3 "Start of Slave Select and End of Slave Select Interrupts"](#)).

## 32.5 Master mode

In Master mode, the device controls the SCK line, and as such, initiates data transfers and determines when any slaves broadcast data onto the SPI bus.

Master mode of this device can be configured in four different modes, configured by the TXR and RXR bits:

- Full-Duplex mode
- Receive Only mode
- Transmit Only mode
- Transfer Off mode

The modes are illustrated in [Table 32-1](#), below:

**TABLE 32-1: MASTER MODE TXR/RXR SETTINGS**

|         | TXR = 1                                                                                                                                                                                                 | TXR = 0                                                                                                                                                                         |
|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RXR = 1 | Full-Duplex mode<br>If BMODE = 1, transfer when RxFIFO is not full and TxFIFO is not empty<br>If BMODE = 0, Transfer when RXFIFO is not full, TXFIFO is not empty, and the Transfer Counter is non-zero | Receive Only mode<br>Transfer when RxFIFO is not full and the Transfer Counter is non-zero<br>Transmitted data is either the top of the FIFO or the most recently received data |
| RXR = 0 | Transmit Only mode<br>If BMODE = 1, transfer when TxTxFIFO is not empty<br>If BMODE = 0, Transfer when TXFIFO is not empty and the Transfer Counter is non-zero<br>Received data is not stored          | No Transfers                                                                                                                                                                    |

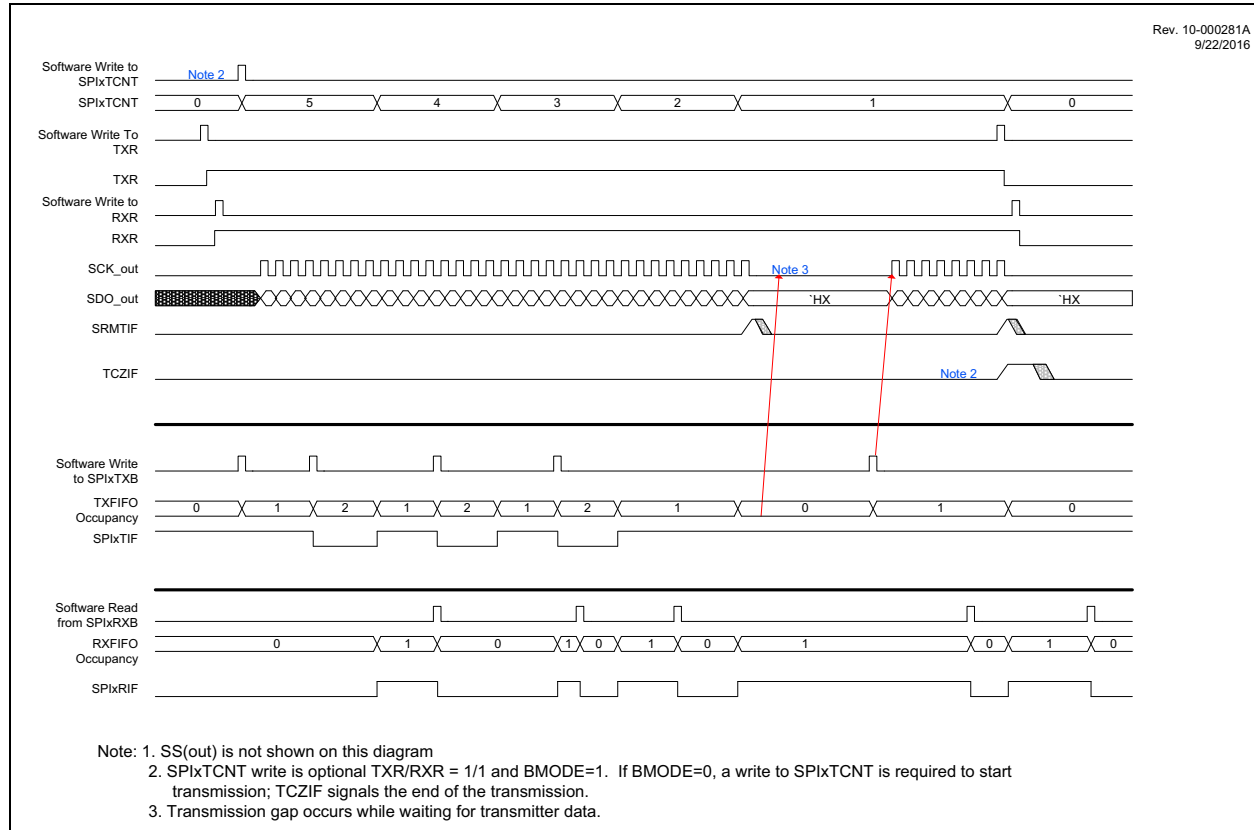
## 32.5.1 FULL-DUPLEX MODE

When both TXR and RXR are set, the SPI master is in Full-Duplex mode. In this mode, data transfer triggering is affected by the BMODE bit of SPIxCON0.

When BMODE = 1, data transfers will occur whenever both the RXFIFO is not full and there is data present in the TXFIFO. In practice, as long as the RXFIFO is not full, data will be transmitted/received as soon as the SPIxTxB register is written to, matching functionality of SPI (MSSP) modules on older 8-bit Microchip devices. The SPIxTCNT will decrement with each transfer. However, when SPIxTCNT is zero the next transfer is not inhibited and the corresponding SPIxTCNT decrement will cause the count to roll over to the maximum value. [Figure 32-3](#) shows an example of a communication using this mode.

When BMODE = 0, the transfer counter (SPIxTCNTH/ SPIxTCNTL) must also be written to before transfers will occur, and transfers will cease when the transfer counter reaches '0'. For example, if SPIxTXB is written twice and then SPIxTCTL is written with '3' then the transfer will start with the SPIxTCTL write. The two bytes in the TXFIFO will be sent after which the transfer will suspend until the third and last byte is written to SPIxTXB.

**FIGURE 32-3: SPI MASTER OPERATION – DATA EXCHANGE, TXR/RXR = 1/1**



## 32.5.2 TRANSMIT ONLY MODE

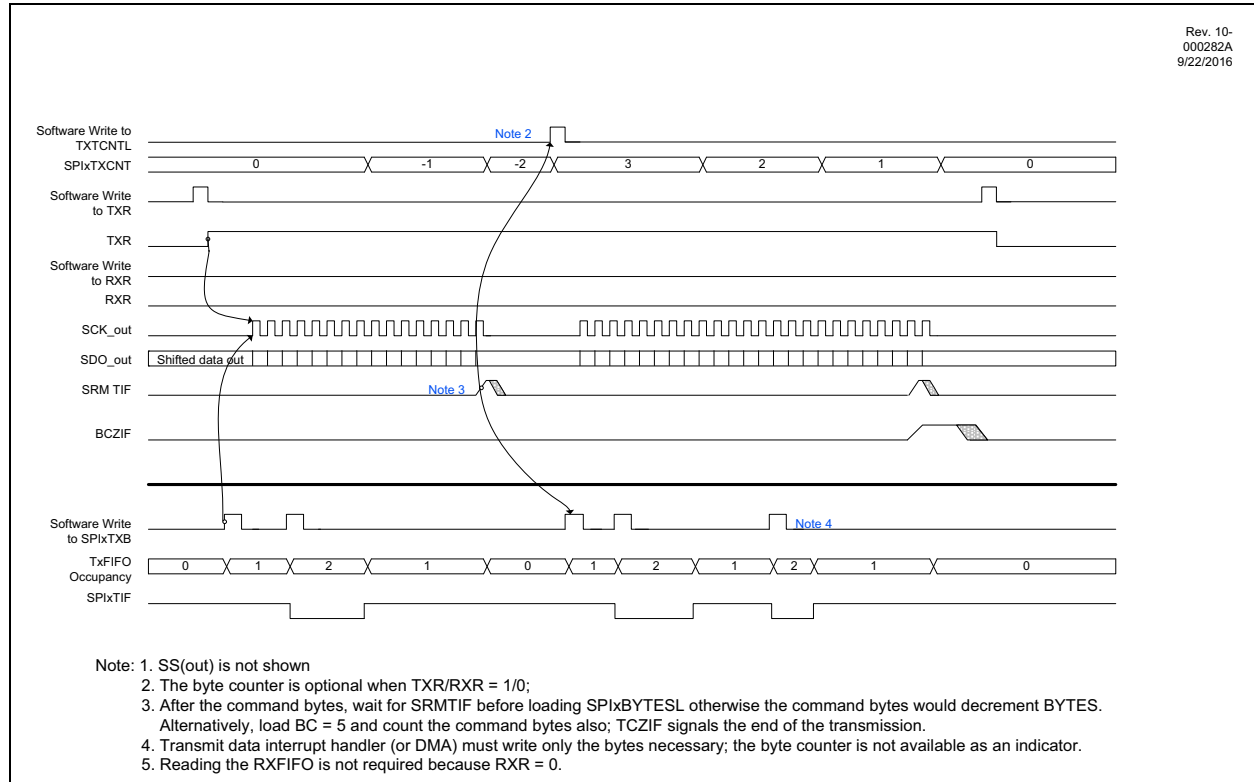
When TXR is set and RXR is clear, the SPI master is in Transmit Only mode. In this mode, data transfer triggering is affected by the BMODE bit of SPIxCON0.

When BMODE = 1, data transfers will occur whenever TXFIFO is not empty. Data will be transmitted as soon as the TXFIFO register is written to, matching functionality of SPI (MSSP) modules on previous 8-bit Microchip devices. The SPIxTCNT will decrement with each transfer. However, when SPIxTCNT is zero the next transfer is not inhibited and the corresponding SPIxTCNT decrement will cause the count to roll over to the maximum value. Any data received in this mode is not stored in RXFIFO. [Figure 32-4](#) shows an example of sending a command and then sending a byte of data, using this mode.

When BMODE = 0, the transfer counter (SPIxTCNTH/L) must also be written to before transfers will occur, and transfers will cease when the transfer counter reaches '0'.

For example, if SPIxTXB is written twice and then SPIxTCTL is written with '3', the transfer will start with the SPIxTCTL write. The two bytes in the TXFIFO will be sent after which the transfer will suspend until the third and last byte is written to SPIxTXB.

**FIGURE 32-4: SPI MASTER OPERATION, COMMAND+WRITE DATA, TXR/RXR=1/0**

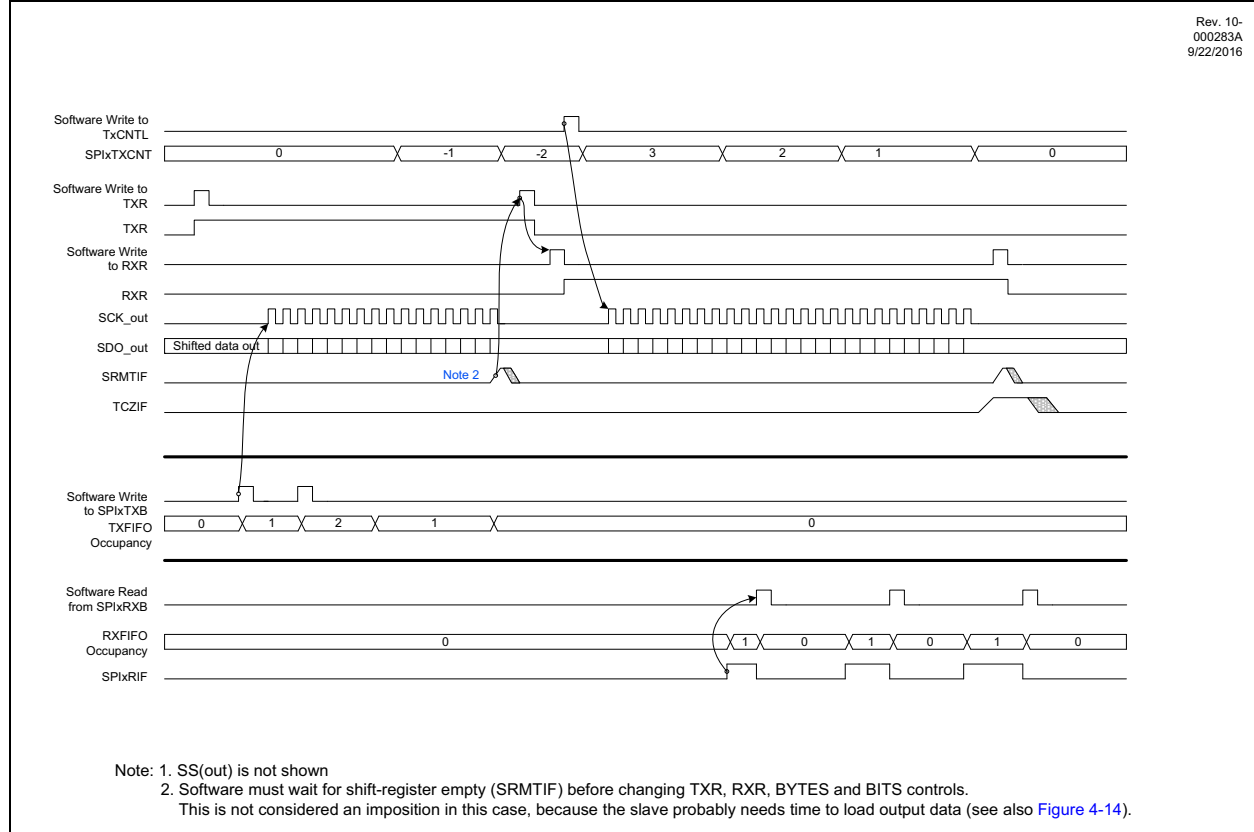


## 32.5.3 RECEIVE ONLY MODE

When RXR is set and TXR is clear, the SPI master is in Receive Only mode. In this mode, data transfers when the RXFIFO is not full and the Transfer Counter is non-zero. In this mode, writing a value to SPIxTCNTL will start the clocks for transfer. The clocks will suspend while the RXFIFO is full and cease when the SPIxTCNT reaches zero (see [Section 32.4 “Transfer Counter”](#)). If there is any data in the TXFIFO, the first

data written to the TXFIFO will be transmitted on each data exchange, although the TXFIFO occupancy will not change, meaning that the same message will be sent on each transmission. If there is no data in the TXFIFO, the most recently received data will instead be transmitted. [Figure 32-5](#) shows an example of sending a command using [Section 32.5.2 “Transmit Only Mode”](#) and then receiving a byte of data using this mode.

**FIGURE 32-5: SPI MASTER OPERATION, COMMAND+READ DATA, TXR/RXR=0/1**



## 32.5.4 TRANSFER OFF MODE

When both TXR and RXR are cleared, the SPI master is in Transfer Off mode. In this mode, SCK will not toggle and no data is exchanged. However, writes to SPIxTXB will be transferred to the TXFIFO which will be transmitted if the TXR bit is set.

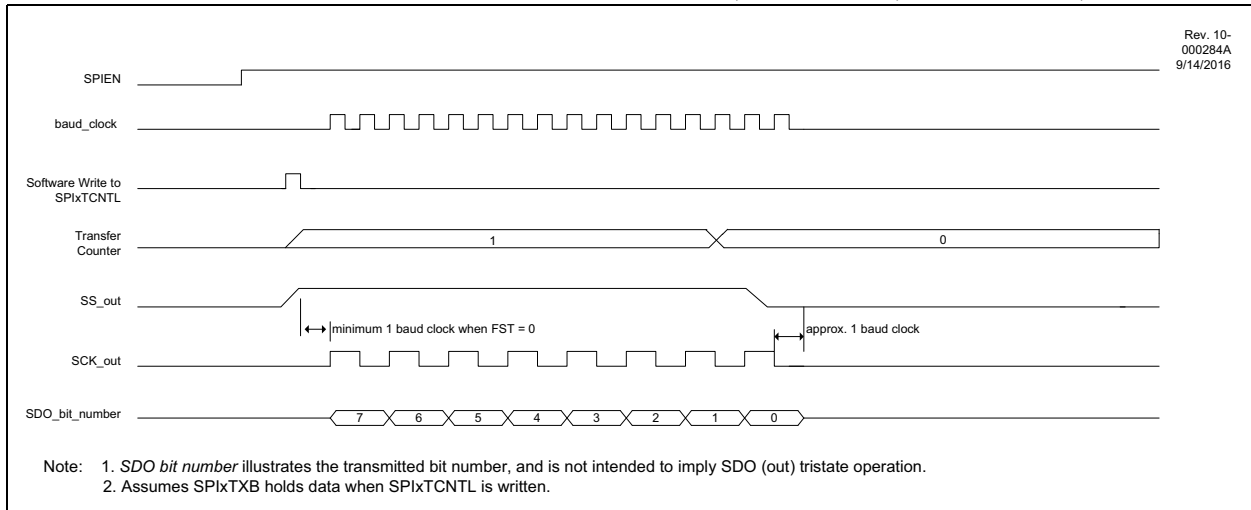
## 32.5.5 MASTER MODE SLAVE SELECT CONTROL

### 32.5.5.1 Hardware Slave Select Control

This SPI module allows for direct hardware control of a Slave Select output. The Slave Select output SS(out) is controlled both directly, through the SSET bit of SPIxCON2, as well indirectly by the hardware while the transfer counter is non-zero (see [Section 32.4 “Transfer Counter”](#)). SS(out) is steered by the PPS registers to pins (see [Section 17.2 “PPS Outputs”](#))

and its polarity is controlled by the SSP bit of SPIxCON1. Setting the SSET bit will also assert SS(out). Clearing the SSET bit will leave SS(out) to be controlled by the Transfer Counter. When the Transfer Counter is loaded, the SPI module will automatically assert the SS. When the Transfer Counter decrements to zero, the SPI module will deassert SS either one baud period after the final SCK pulse of the final transfer (if CKE/SMP = 0/1) or one half baud period otherwise (see [Figure 32-6](#)).

**FIGURE 32-6: SPI MASTER SS OPERATION- CKE = 0, BMODE = 1, TCWIDTH = 0, SSP = 0**



### 32.5.5.2 Software Slave Select Control

Slave Select can also be controlled through software via a general purpose I/O pin. In this case, ensure that the pin in question is configured as a GPIO through PPS (see [Section 17.2 “PPS Outputs”](#)), and ensure that the pin is set as an output (clear the appropriate bit in the appropriate TRIS register). In this case, SSET will not affect the Slave Select, the Transfer Counter will not automatically control the Slave Select output, and all setting and clearing of the Slave Select output line must be directly controlled by software.

## 32.5.6 MASTER MODE SPI CLOCK CONFIGURATION

### 32.5.6.1 SPI Clock Selection

The clock source for SPI Master modes is selected by the SPIxCLK register. Selections include the following:

- FOSC
- HFINTOSC
- CLKREF
- Timer0\_overflow
- Timer2\_Postscaled
- Timer4\_Postscaled
- Timer6\_Postscaled
- SMT\_match

The SPIxBAUD register allows for dividing this clock. The frequency of the SCK output is defined by [Equation 32-1](#):

#### EQUATION 32-1: FREQUENCY OF SCK OUTPUT SIGNAL

$$F_{\text{BAUD}} = \frac{F_{\text{CSEL}}}{(2 \cdot (\text{BAUD} + 1))}$$

where FBAUD is the baud rate frequency output on the SCK pin, FCSEL is the frequency of the input clock selected by the SPIxCLK register, and BAUD is the value contained in the SPIxBAUD register.

### 32.5.6.2 CKE, CKP and SMP

The CKP, CKE, and SMP bits control the relationship between the SCK clock output, SDO output data changes, and SDI input data sampling. The bit functions are as follows:

- CKP - SCK output polarity
- CKE - SDO output change relative to the SCK clock
- SMP - SDI input sampling relative to the clock edges

The CKE bit, when set, inverts the low Idle state of the SCK output to a high Idle state.

[Figure 32-7](#) through [Figure 32-10](#) illustrate the eight possible combinations of the CKP, CKE, and SMP bit selections.

When the CKE bit is set, the SDO data is valid before there is a clock edge on SCK. When the CKE bit is cleared, the SDO data is undefined prior to the first SCK edge.

**Note:** All timing diagrams assume the LSBF bit of SPIxCON0 is cleared.

**FIGURE 32-7: CLOCKING DETAIL – MASTER MODE, CKE/SMP = 0/0**

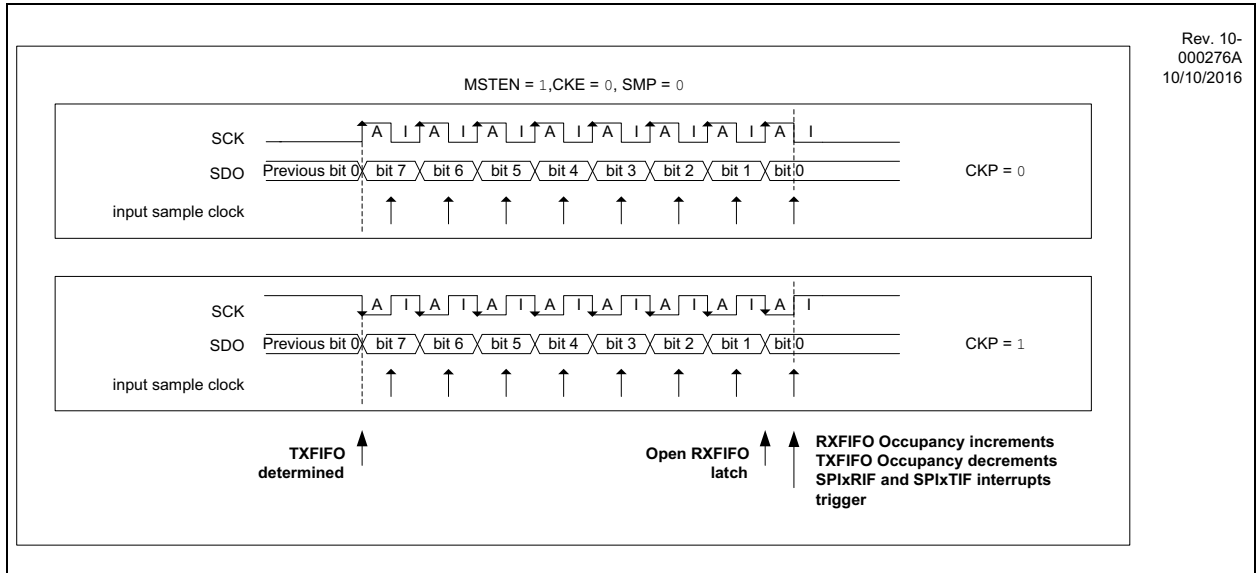


**FIGURE 32-8: CLOCKING DETAIL – MASTER MODE, CKE/SMP = 1/1**

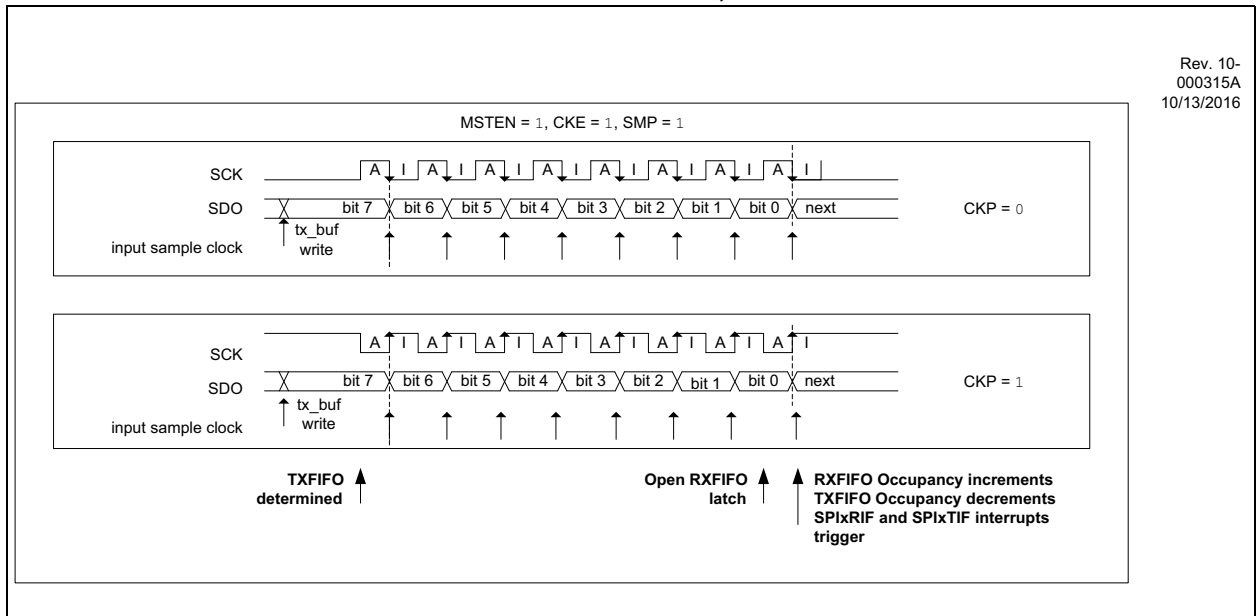




**FIGURE 32-9: CLOCKING DETAIL – MASTER MODE, CKE = 0, SMP = 1**

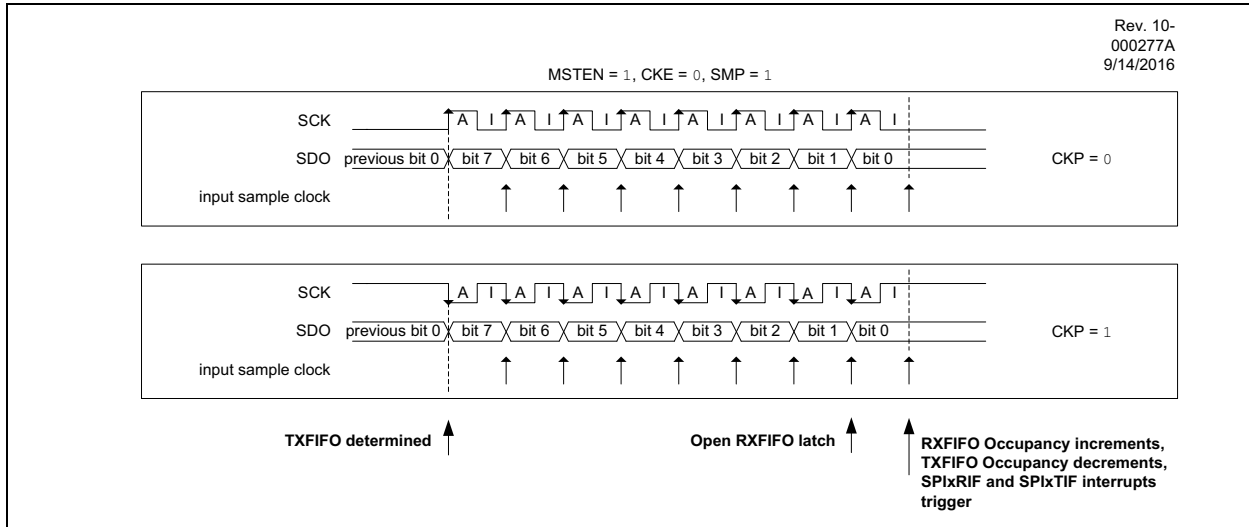
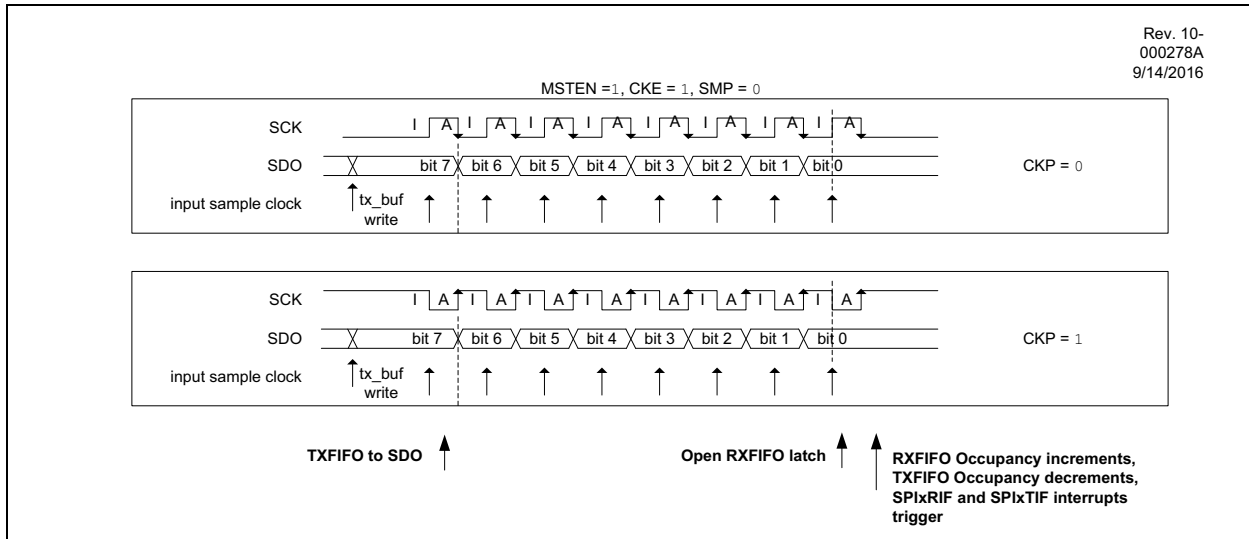


**FIGURE 32-10: CLOCKING DETAIL – MASTER MODE, CKE = 1, SMP = 0**



### 32.5.6.3 SCK Start-Up Delay

When starting an SPI data exchange, the master device sets the SS output (either through hardware or software) and then triggers the module to send data. These data triggers are synchronized to the clock selected by the SPIxCLK register before the first SCK pulse appears, usually requiring one or two clocks of the selected clock.

The SPI module includes synchronization delays on SCK generation specifically designed to ensure that the Slave Select output timing is correct, without requiring precision software timing loops.

When the value of the SPIxBAUD register is a small number (indicating higher SCK frequencies), the synchronization delay can be relatively long between setting SS and the first SCK. With larger values of

SPIxBAUD (indicating lower SCK frequencies), this delay is much smaller and the first SCK can appear relatively quickly after SS is set.

By default, the SPI module inserts a ½ baud delay (half of the period of the clock selected by the SPIxCLK register) before the first SCK pulse. This allows for systems with a high SPIxBAUD value to have extra set-up time before the first clock. Setting the FST bit in SPIxCON1 removes this additional delay, allowing systems with low SPIxBAUD values (and thus, long synchronization delays) to forego this unnecessary extra delay.

## 32.6 Slave Mode

### 32.6.1 SLAVE MODE TRANSMIT OPTIONS

The SDO output of the SPI module in Slave mode is controlled by the TXR bit of SPIxCON2, the TRIS bit associated with the SDO pin, the Slave Select input, and the current state of the TXFIFO. This control is summarized in Table 32-2. In this table, TRISxn refers to the bit in the TRIS register corresponding to the pin that SDO has been assigned with PPS, TXR is the Transmit Data Required Control bit of SPIxCON2, SS is the state of the Slave Select input, and TXBE is the TXFIFO Buffer Empty bit of SPIxSTATUS.

#### 32.6.1.1 SDO Drive/Tri-state

The TRIS bit associated with the SDO pin controls whether the SDO pin will tri-state. When this TRIS bit is cleared, the pin will always be driving to a level, even when the SPI module is inactive. When the SPI module is inactive (either due to the master not clocking the SCK line or the SS being false), the SDO pin will be driven to the value of the LAT bit associated with the SDO pin. When the SPI module is active, its output is determined by both TXR and whether there is data in the TXFIFO.

When the TRIS bit associated with the SDO pin is set, the pin will only have an output level driven to it when TXR = 1 and the Slave Select input is true. In all other cases, the pin will be tri-stated.

#### 32.6.1.2 SDO Output Data

The TXR bit controls the nature of the data that is transmitted in Slave mode. When TXR is set, transmitted data is taken from the TXFIFO. If the FIFO is empty, the most recently received data will be transmitted and the TXUIF flag will be set to indicate that a transmit FIFO underflow has occurred.

When TXR is cleared, the data will be taken from the TXFIFO, and the TXFIFO occupancy will not decrease. If the TXFIFO is empty, the most recently received data will be transmitted, and the TXUIF bit will not be set. However, if the TRIS bit associated with the SDO pin is set, clearing the TXR bit will cause the SPI module to not output any data to the SDO pin.

**TABLE 32-2: SLAVE MODE TRANSMIT**

| TRISxn <sup>(1)</sup> | TXR | SS    | TXBE | SDO State                                                                                                           |
|-----------------------|-----|-------|------|---------------------------------------------------------------------------------------------------------------------|
| 0                     | 0   | FALSE | 0    | Drives state determined by LATxn(2)                                                                                 |
| 0                     | 0   | FALSE | 1    | Drives state determined by LATxn(2)                                                                                 |
| 0                     | 0   | TRUE  | 0    | Outputs the oldest byte in the TXFIFO<br>Does not remove data from the TXFIFO                                       |
| 0                     | 0   | TRUE  | 1    | Outputs the most recently received byte                                                                             |
| 0                     | 1   | FALSE | 0    | Drives state determined by LATxn(2)                                                                                 |
| 0                     | 1   | FALSE | 1    | Drives state determined by LATxn(2)                                                                                 |
| 0                     | 1   | TRUE  | 0    | Outputs the oldest byte in the TXFIFO<br>Removes transmitted byte from the TXFIFO<br>Decrements occupancy of TXFIFO |
| 0                     | 1   | TRUE  | 1    | Outputs the most recently received byte<br>Sets the TXUIF bit of SPIxINTF                                           |
| 1                     | 0   | FALSE | 0    | Tri-stated                                                                                                          |
| 1                     | 0   | FALSE | 1    | Tri-stated                                                                                                          |
| 1                     | 0   | TRUE  | 0    | Tri-stated                                                                                                          |
| 1                     | 0   | TRUE  | 1    | Tri-stated                                                                                                          |
| 1                     | 1   | FALSE | 0    | Tri-stated                                                                                                          |
| 1                     | 1   | FALSE | 1    | Tri-stated                                                                                                          |
| 1                     | 1   | TRUE  | 0    | Outputs the oldest byte in the TXFIFO<br>Removes transmitted byte from the TXFIFO<br>Decrements occupancy of TXFIFO |
| 1                     | 1   | TRUE  | 1    | Outputs the most recently received byte<br>Sets the TXUIF bit of SPIxINTF                                           |

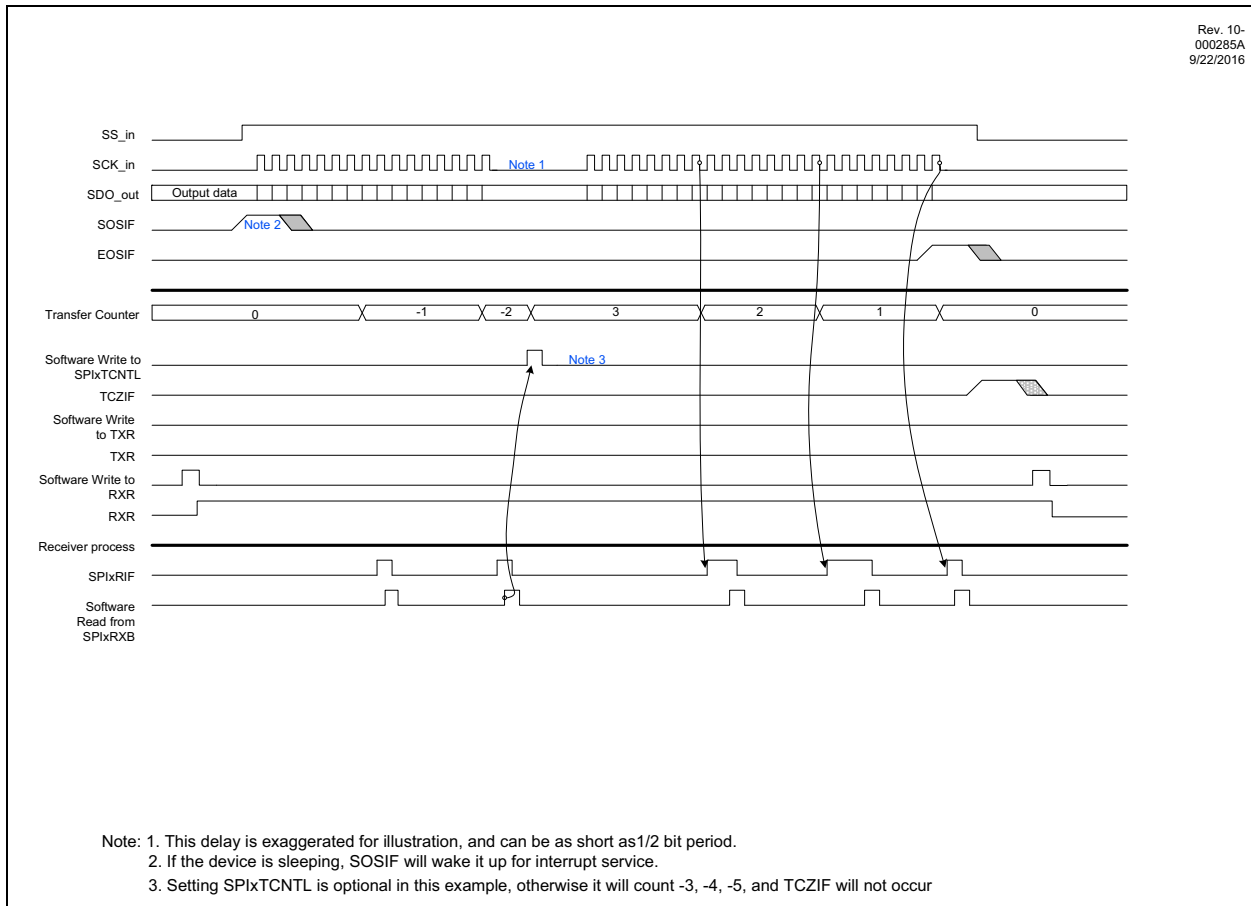
**Note 1:** TRISxn is the bit in the TRISx register corresponding to the pin that SDO has been assigned with PPS.

**Note 2:** LATxn is the bit in the LATx register corresponding to the pin that SDO has been assigned with PPS.

## 32.6.2 SLAVE MODE RECEIVE OPTIONS

The RXR bit controls the nature of receptions in Slave mode. When RXR is set, the SDI input data will be stored in the RXFIFO if it is not full. If the RXFIFO is full, the RXOIF bit will be set to indicate an RXFIFO overflow error and the data is discarded. When RXR is cleared, all received data will be ignored and not stored in the RXFIFO (although it may still be used for transmission if TXFIFO is empty). Figure 32-11 shows a typical Slave mode communication, showing a case where the master writes two then three bytes, showing interrupts as well as the behavior of the transfer counter in Slave mode (see Section 32.4.3 “Transfer Counter in Slave mode” for more details on the transfer counter in Slave mode as well as Section 32.8 “SPI Interrupts” for more information on interrupts).

**FIGURE 32-11: SPI SLAVE MODE OPERATION – INTERRUPT-DRIVEN, MASTER WRITES 2+3 BYTES**



## 32.6.3 SLAVE MODE SLAVE SELECT

In Slave mode, an external Slave Select Signal can be used to synchronize communication with the master device. The Slave Select line is held in its inactive state (high by default) until the master device is ready to communicate. When the Slave Select transitions to its active state, the slave knows that a new transmission is starting.

When the Slave Select goes false at the end of the transmission the receive function of the selected SPI slave device returns to the inactive state. The slave is then ready to receive a new transmission when the Slave Select goes True again.

The Slave Select signal is received on the  $\overline{SS}$  input pin. This pin is remappable with the SPIxSSPPS register (see [Section 17.1 “PPS Inputs”](#)). When the input on this pin is true, transmission and reception are enabled, and the SDO pin is driven. When the input on this pin is false, the SDO pin is either tri-stated (if the TRIS bit associated with the SDO pin is set) or driven to the value of the LAT bit associated with the SDO pin (if the TRIS bit associated with the SDO pin is cleared). In addition, the SCK input is ignored.

If the SS input goes False, while a data transfer is still in progress, it is considered a Slave Select fault. The SSFLT bit of SPIxCON2 indicates whether such an event has occurred. The transfer counter value determines the number of bits in a valid data transfer (see [Section 32.4 “Transfer Counter”](#) for more details).

The Slave Select polarity is controlled by the SSP bit of SPIxCON1. When SSP is set (its default state), the Slave Select input is active-low, and when it is cleared, the Slave Select input is active-high.

The Slave Select for the SPI module is controlled by the SSET bit of SPIxCON2. When the bit is cleared (its default state), the Slave Select will act as described above. When the bit is set, the SPI module will behave as if the SS input was always in its active state.

**Note:** When SSET is set, the effective SS(in) signal is always active. Hence, the SSFLT bit may be disregarded.

## 32.6.4 SLAVE MODE CLOCK CONFIGURATION

In Slave mode, SCK is an input, and must be configured to the same polarity and clock edge as the master device. As in Master mode, the polarity of the clock input is controlled by the CKP bit of SPIxCON1 and the clock edge used for transmitting data is controlled by the CKE bit of SPIxCON1.

## 32.6.5 DAISY-CHAIN CONFIGURATION

The SPI bus can be connected in a daisy-chain configuration. The first slave output is connected to the second slave input, the second slave output is connected to the third slave input, and so on. The final slave output is connected to the master input. Each slave sends out, during a second group of clock pulses, an exact copy of what was received during the first group of clock pulses. The whole chain acts as one large communication shift register. The daisy-chain feature only requires a single Slave Select line from the master device connected to all slave devices (alternately, the slave devices can be configured to ignore the Slave Select line by setting the SSET bit). In a typical Daisy-Chain configuration, the SCK signal from the master is connected to each of the slave device SCK inputs. However, the SCK input and output are separate signals selected by the PPS control. When the PPS selection is made to configure the SCK input and SCK output on separate pins then, the SCK output will follow the SCK input, allowing for SCK signals to be daisy-chained like the SDO/SDI signals.

[Figure 32-12](#) shows the block diagram of a typical daisy-chain connection, and [Figure 32-13](#) shows the block diagram of a daisy-chain connection possible using this SPI module.

**FIGURE 32-12: TRADITIONAL SPI DAISY – CHAIN CONNECTION**

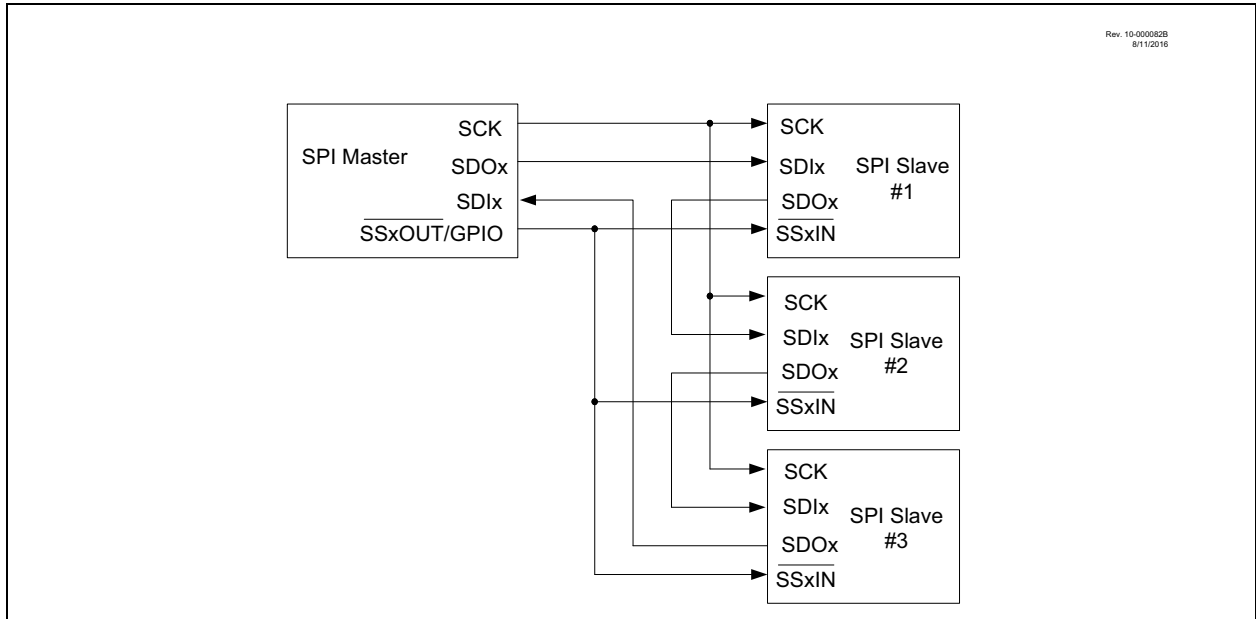
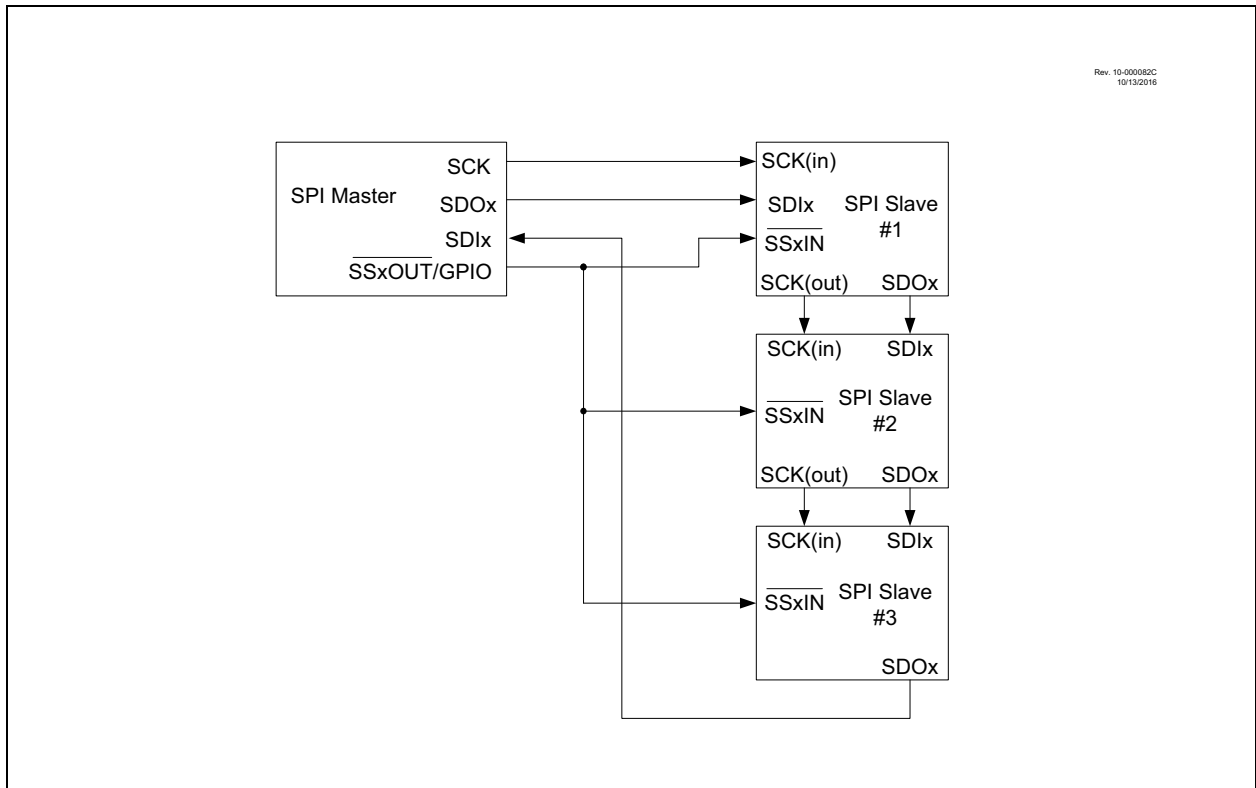


**FIGURE 32-13: SPI DAISY-CHAIN CONNECTION WITH CHAINED SCK**



## 32.7 SPI Operation in Sleep Mode

SPI Master mode will operate in Sleep, provided the clock source selected by SPIxCLK is active in Sleep mode. FIFOs will operate as they would when the part is awake. When TXR = 1, the TXFIFO will need to contain data in order for transfers to take place in Sleep. All interrupts will still set the interrupt flags in Sleep but only enabled interrupts will wake the device from Sleep.

SPI Slave mode will operate in Sleep, because the clock is provided by an external master device. FIFOs will still operate and interrupts will set interrupt flags, and enabled interrupts will wake the device from Sleep.

## 32.8 SPI Interrupts

There are three top level SPI interrupts in the PIRx register:

- SPI Transmit
- SPI Receive
- SPI Module status

The status interrupts are enabled at the module level in the SPIxINTE register. Only enabled status interrupts will cause the single top level SPIxIF flag to be set.

### 32.8.1 SPI RECEIVER DATA INTERRUPT

The SPI Receiver Data Interrupt is set when RXFIFO contains data, and is cleared when the RXFIFO is empty. The interrupt flag SPI1RXIF is located in PIRx and the interrupt enable SPI1RXIE is located in PIEx. This interrupt flag is read-only.

### 32.8.2 SPI TRANSMITTER DATA INTERRUPT

The SPI Transmitter Data Interrupt is set when TXFIFO is not full, and is cleared when the TXFIFO is full. The interrupt flag SPI1TXIF is located in PIRx and the interrupt enable SPI1TXIE is located in PIEx. The interrupt flag is read-only.

### 32.8.3 SPI MODULE STATUS INTERRUPTS

The SPIxIF flag in the respective PIR register is set when any of the individual status flags in SPIxINTF and their respective SPIxINTE bits are set. In order for the setting of any specific interrupt flag to interrupt normal program flow both the SPIxIE bit as well as the specific bit in SPIxINTE associated with that interrupt must be set.

The Status Interrupts are:

- Shift Register Empty Interrupt
- Transfer Counter is Zero Interrupt
- Start of Slave Select Interrupt
- End of Slave Select Interrupt
- Receiver Overflow Interrupt
- Transmitter Underflow Interrupt

## 32.8.3.1 Shift Register Empty Interrupt

The Shift Register Empty interrupt flag and enable are the SRMTIF and SRMTIE bits, respectively. This interrupt is only available in Master mode and triggers when a data transfer completes and conditions are not present to start a new transfer, as dictated by the TXR and RXR bits (see Table 32-1 for conditions for starting a new Master mode data transfer with different TXR/RXR settings). This interrupt will be triggered at the end of the last full bit period, after SCK has been low for one ½-baud period. See Figure 32-14 for more details of the timing of this interrupt as well as other interrupts. This bit will not clear itself when the conditions for starting a new transfer occur, and must be cleared in software.

## 32.8.3.2 Transfer Counter is Zero Interrupt

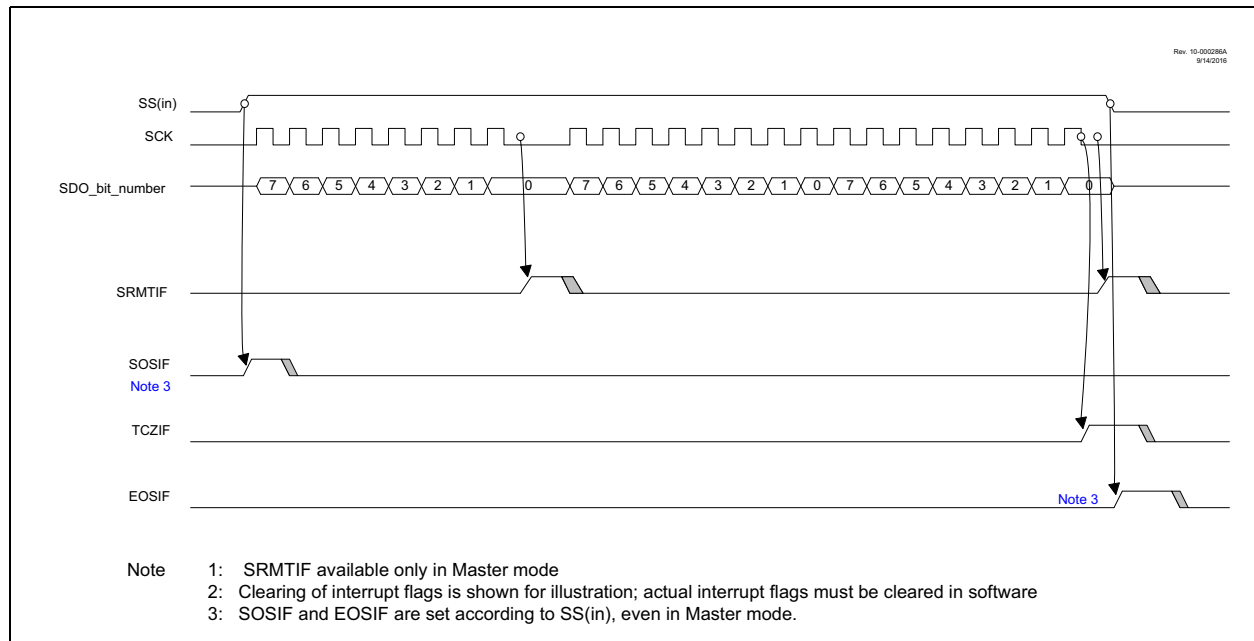
The transfer counter is zero interrupt flag and enable are the TCZIF and TCZIE bits, respectively. This interrupt will trigger when the transfer counter (defined by BMODE, SPIxTCTH/L and SPIxTWIDTH) decrements from one to zero. See Figure 32-14 for more details on the timing of this interrupt as well as other interrupts. This bit must be cleared in software.

**Note:** The TCZIF flag only indicates that the transfer counter has decremented from one to zero, and may not indicate that the entire data transfer process is complete. Either poll the BUSY bit of SPIxCON2 and wait for it to be cleared or use the Shift Register Empty Interrupt (SRMTIF) to determine if a data transfer is fully complete.

## 32.8.3.3 Start of Slave Select and End of Slave Select Interrupts

The start of Slave Select interrupt flag and enable are the SOSIF and SOSIE bits, respectively, and the end of Slave Select interrupt flag and enable are similarly designated by the EOSIF and EOSIE bits. These interrupts trigger at the leading and trailing edges of the Slave Select input. Note that the interrupts are active in both master and Slave mode, and will trigger on transitions of the Slave Select input regardless of which mode the SPI is in. In Master mode, PPS should be used to route the Slave Select input to the same pin as the Slave Select output, allowing these interrupts to trigger on changes to the Slave Select output. Also note that in Slave mode, changing the SSET bit can trigger these interrupts, as it changes the effective input value of Slave Select. Both SOSIF and EOSIF must be cleared in software.

**FIGURE 32-14: TRANSFER AND SLAVE SELECT INTERRUPT TIMINGS**



## 32.8.3.4 Receiver Overflow and Transmitter Underflow Interrupts

The receiver overflow interrupt triggers if data is received when the RXFIFO is already full and RXR = 1. In this case, the data will be discarded and the RXOIF bit will be set. The receiver overflow interrupt flag is the RXOIF bit of SPIxINTF. The receiver overflow interrupt enable bit is the RXOIE bit of SPIxINTE.

The Transmitter Underflow interrupt flag triggers if a data transfer begins when the TXFIFO is empty and TXR = 1. In this case, the most recently received data will be transmitted and the TXUIF bit will be set. The transmitter underflow interrupt flag is the TXUIF bit of SPIxINTF. The transmitter underflow interrupt enable bit is the TXUIE bit of SPIxINTE.

Both of these interrupts will only occur in Slave mode, as Master mode will not allow the RXFIFO to overflow or the TXFIFO to underflow.



## 32.9 Register definitions: SPI

### REGISTER 32-1: SPIxINTF: SPI INTERRUPT FLAG REGISTER

| R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | U-0 | R/W/HS-0/0 | R/W/HS-0/0 | U-0   |
|------------|------------|------------|------------|-----|------------|------------|-------|
| SRMTIF     | TCZIF      | SOSIF      | EOSIF      | —   | RXOIF      | TXUIF      | —     |
| bit 7      |            |            |            |     |            |            | bit 0 |

#### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

HS = Bit can be set by hardware

bit 7 **SRMTIF**: Shift Register Empty Interrupt Flag bit

#### Slave mode:

This bit is ignored

#### Master mode:

1 = The data transfer is complete

0 = Either no data transfers have occurred or a data transfer is in progress

bit 6 **TCZIF**: Transfer Counter is Zero Interrupt Flag bit

1 = The transfer counter (as defined by BMODE in [Register 32-7](#), TCNTH/L, and TWIDTH) has decremented to zero

0 = No interrupt pending

bit 5 **SOSIF**: Start of Slave Select Interrupt Flag bit

1 = SS(in) transitioned from false to true

0 = No interrupt pending

bit 4 **EOSIF**: End of Slave Select Interrupt Flag bit

1 = SS(in) transitioned from true to false

0 = No interrupt pending

bit 3 **Unimplemented**: Read as '0'

bit 2 **RXOIF**: Receiver Overflow Interrupt Flag bit

1 = Data transfer completed when RXBF = 1 (edge triggered) and RXR = 1

0 = No interrupt pending

bit 1 **TXUIF**: Transmitter Underflow Interrupt Flag bit

1 = Slave Data transfer started when TXBE = 1 and TXR = 1

0 = No interrupt pending

bit 0 **Unimplemented**: Read as '0'

## REGISTER 32-2: SPIxINTE: SPI INTERRUPT ENABLE REGISTER

|         |         |         |         |     |         |         |       |
|---------|---------|---------|---------|-----|---------|---------|-------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | U-0 | R/W-0/0 | R/W-0/0 | U-0   |
| SRMTIE  | TCZIE   | SOSIE   | EOSIE   | —   | RXOIE   | TXUIE   | —     |
| bit 7   |         |         |         |     |         |         | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

- bit 7 **SRMTIE**: Shift Register Empty Interrupt Enable bit  
 1 = Enables the Shift Register Empty Interrupt  
 0 = Disables the Shift Register Empty Interrupt
- bit 6 **TCZIE**: Transfer Counter is Zero Interrupt Enable bit  
 1 = Enables the Transfer Counter is Zero Interrupt  
 0 = Disables the Transfer Counter is Zero Interrupt
- bit 5 **SOSIE**: Start of Slave Select Interrupt Enable bit  
 1 = Enables the Start of Slave Select Interrupt  
 0 = Disables the Start of Slave Select Interrupt
- bit 4 **EOSIE**: End of Slave Select Interrupt Enable bit  
 1 = Enables the End of Slave Select Interrupt  
 0 = Disables the End of Slave Select Interrupt
- bit 3 **Unimplemented**: Read as '0'
- bit 2 **RXOIE**: Receiver Overflow Interrupt Enable bit  
 1 = Enables the Receiver Overflow Interrupt  
 0 = Disables the Receiver Overflow Interrupt
- bit 1 **TXUIE**: Transmitter Underflow Interrupt Enable bit  
 1 = Enables the Transmitter Underflow Interrupt  
 0 = Disables the Transmitter Underflow Interrupt
- bit 0 **Unimplemented**: Read as '0'

## REGISTER 32-3: SPIxTCNTL – SPI TRANSFER COUNTER LSB REGISTER

|         |         |         |         |         |         |         |         |
|---------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| TCNT7   | TCNT6   | TCNT5   | TCNT4   | TCNT3   | TCNT2   | TCNT1   | TCNT0   |
| bit 7   |         |         |         |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

- bit 7-0 **TCNT<7:0>**:  
**BMODE = 0**  
 Bits 10-3 of the Transfer Counter, counting the total number of bits to transfer  
**BMODE = 1**  
 Bits 7-0 of the Transfer Counter, counting the total number of bytes to transfer

**Note:** This register should not be written to while a transfer is in progress (BUSY bit of SPIxCON2 is set).

## REGISTER 32-4: SPIxTCNTH: SPI TRANSFER COUNTER MSB REGISTER

|       |     |     |     |     |         |         |         |
|-------|-----|-----|-----|-----|---------|---------|---------|
| U-0   | U-0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| —     | —   | —   | —   | —   | TCNT10  | TCNT9   | TCNT8   |
| bit 7 |     |     |     |     | bit 0   |         |         |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

bit 7-3 **Unimplemented:** Read as '0'

bit 2-0 **TCNT<10:8>:**

BMODE = 0

Bits 13-11 of the Transfer Counter, counting the total number of bits to transfer

BMODE = 1

Bits 10-8 of the Transfer Counter, counting the total number of bytes to transfer

**Note:** This register should not be written to while a transfer is in progress (BUSY bit of SPIxCON2 is set).

## REGISTER 32-5: SPIxTWIDTH: SPI TRANSFER WIDTH REGISTER

|       |     |     |     |     |         |         |         |
|-------|-----|-----|-----|-----|---------|---------|---------|
| U-0   | U-0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| —     | —   | —   | —   | —   | TWIDTH2 | TWIDTH1 | TWIDTH0 |
| bit 7 |     |     |     |     | bit 0   |         |         |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

bit 7-3 **Unimplemented:** Read as '0'

bit 2-0 **TWIDTH<2:0>:**

BMODE = 0

Bits 2-0 of the Transfer Counter, counting the total number of bits to transfer

BMODE = 1

Size (in bits) of each transfer counted by the transfer counter

111 = 7 bits

110 = 6 bits

101 = 5 bits

100 = 4 bits

011 = 3 bits

010 = 2 bits

001 = 1 bit

000 = 8 bits

**Note:** This register should not be written to while a transfer is in progress (BUSY bit of SPIxCON2 is set).

## REGISTER 32-6: SPIxBAUD: SPI BAUD RATE REGISTER

|         |         |         |         |         |         |         |         |
|---------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| BAUD7   | BAUD6   | BAUD5   | BAUD4   | BAUD3   | BAUD2   | BAUD1   | BAUD0   |
| bit 7   |         |         |         |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

bit 7-0 **BAUD<7:0>**: Baud Clock Prescaler Select bits

SCK high or low time:  $TSC = SPI \text{ Clock Period} * (BAUD + 1)$

SCK toggle frequency:  $FSCK = FBAUD = SPI \text{ Clock Frequency} / (2 * (BAUD + 1))$

**Note:** This register should not be written while the SPI is enabled (EN bit of SPIxCON0 = 1)

## REGISTER 32-7: SPIxCON0: SPI CONFIGURATION REGISTER 0

|         |     |     |     |     |         |         |         |
|---------|-----|-----|-----|-----|---------|---------|---------|
| R/W-0/0 | U-0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| EN      | —   | —   | —   | —   | LSBF    | MST     | BMODE   |
| bit 7   |     |     |     |     |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

bit 7 **EN**: SPI Module Enable Control bit

1 = SPI is enabled

0 = SPI is disabled,

bit 6-3 **Unimplemented**: Read as '0'

bit 2 **LSBF**: LSb-First Data Exchange bit

1 = Data is exchanged LSb first

0 = Data is exchanged MSb first (traditional SPI operation)

bit 1 **MST**: SPI Operating Mode Master Select bit

1 = SPI module operates as the bus master

0 = SPI module operates as a bus slave

bit 0 **BMODE**: Bit-Length Mode Select bit

1 = SPIxTWIDTH setting applies to every byte: total bits sent is SPIxTWIDTH \* SPIxTCNT, end-of-packet occurs when SPIxTCNT = 0

0 = SPIxTWIDTH setting applies only to the last byte exchanged; total bits sent is SPIxTWIDTH + (SPIxTCNT \* 8)

**Note:** This register should only be written when the EN bit is cleared, or to clear the EN bit.

## REGISTER 32-8: SPIxCON1: SPI CONFIGURATION REGISTER 1

|         |         |         |         |     |         |         |         |
|---------|---------|---------|---------|-----|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | U-0 | R/W-1/1 | R/W-0/0 | R/W-0/0 |
| SMP     | CKE     | CKP     | FST     | —   | SSP     | SDIP    | SDOP    |
| bit 7   |         |         |         |     |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

- bit 7 **SMP:** SPI Input Sample Phase Control bit  
Slave mode:  
 1 = Reserved  
 0 = SDI input is sampled in the middle of data output time  
Master mode:  
 1 = SDI input is sampled at the end of data output time  
 0 = SDI input is sampled in the middle of data output time
- bit 6 **CKE:** Clock Edge Select bit  
 1 = Output data changes on transition from active to idle clock state  
 0 = Output data changes on transition from idle to active clock state
- bit 5 **CKP:** Clock Polarity Select bit  
 1 = Idle state for SCK is high level  
 0 = Idle state for SCK is low level
- bit 4 **FST:** Fast Start Enable bit  
Slave mode:  
 This bit is ignored  
Master mode:  
 1 = Delay to first SCK may be less than ½ baud period  
 0 = Delay to first SCK will be at least ½ baud period
- bit 3 **Unimplemented:** Read as '0'
- bit 2 **SSP:** SS Input/Output Polarity Control bit  
 1 = SS is active-low  
 0 = SS is active-high
- bit 1 **SDIP:** SDI Input Polarity Control bit  
 1 = SDI input is active-low  
 0 = SDI input is active-high
- bit 0 **SDOP:** SDO Output Polarity Control bit  
 1 = SDO output is active-low  
 0 = SDO output is active-high

## REGISTER 32-9: SPIxCON2: SPI CONFIGURATION REGISTER 2

|       |       |     |     |     |         |                    |                    |
|-------|-------|-----|-----|-----|---------|--------------------|--------------------|
| R-0/0 | R-0/0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0            | R/W-0/0            |
| BUSY  | SSFLT | —   | —   | —   | SSET    | TXR <sup>(1)</sup> | RXR <sup>(1)</sup> |
| bit 7 |       |     |     |     |         | bit 0              |                    |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

bit 7 **BUSY:** SPI Module Busy Status bit

1 = Data exchange is busy

0 = Data exchange is not taking place

bit 6 **SSFLT:** SS(in) Fault Status bit

If SSET = 0

1 = SS(in) ended the transaction unexpectedly, and the data byte being received was lost

0 = SS(in) ended normally

If SSET = 1

This bit is unchanged.

bit 5-3 **Unimplemented:** Read as '0'

bit 2 **SSET:** Slave Select Enable bit

Master mode:

1 = SS(out) is driven to the active state continuously

0 = SS(out) is driven to the active state while the transmit counter is not zero

Slave mode:

1 = SS(in) is ignored and data is clocked on all SCK(in) (as though SS = TRUE at all times)

0 = SS(in) enables/disables data input and tri-states SDO if the TRIS bit associated with the SDO pin is set (see [Table 32-2](#) for details)

bit 1 **TXR:** Transmit Data-Required Control bit<sup>(1)</sup>

1 = TxFIFO data is required for a transfer

0 = TxFIFO data is not required for a transfer

bit 0 **RXR:** Receive FIFO Space-Required Control bit<sup>(1)</sup>

1 = Data transfers are suspended if the RxFIFO is full

0 = Received data is not stored in the FIFO

**Note 1:** See [Table 32-1](#) as well as [Section 32.5 “Master mode”](#) and [Section 32.6 “Slave Mode”](#) for more details pertaining to TXR and RXR function.

**2:** This register should not be written to while a transfer is in progress (BUSY bit of SPIxCON2 is set).

## REGISTER 32-10: SPIxSTATUS: SPI STATUS REGISTER

|            |     |       |     |            |       |     |       |
|------------|-----|-------|-----|------------|-------|-----|-------|
| R/C/HS-0/0 | U-0 | R-1/1 | U-0 | R/C/HS-0/0 | S-0/0 | U-0 | R-0/0 |
| TXWE       | —   | TXBE  | —   | RXRE       | CLRBF | —   | RXBF  |
| bit 7      |     |       |     |            |       |     | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

C = Clearable bit

S = Settable bit

HS = Bit can be set by hardware

- bit 7 **TXWE**: Transmit Buffer Write Error bit  
 1 = SPIxTxB was written while TxFIFO was full  
 0 = No error has occurred
- bit 6 **Unimplemented**: Read as '0'
- bit 5 **TXBE**: Transmit Buffer Empty bit (read-only)  
 1 = Transmit buffer TxFIFO is empty  
 0 = Transmit buffer is not empty
- bit 4 **Unimplemented**: Read as '0'
- bit 3 **RXRE**: Receive Buffer Read Error bit  
 1 = SPIxRB was read while RxFIFO was empty  
 0 = No error has occurred
- bit 2 **CLRBF**: Clear Buffer Control bit (write only)  
 1 = Reset the receive and transmit buffers, making both buffers empty  
 0 = Take no action
- bit 1 **Unimplemented**: Read as '0'
- bit 0 **RXBF**: Receive Buffer Full bit (read-only)  
 1 = Receive buffer is full  
 0 = Receive buffer is not full

## REGISTER 32-11: SPIxRxB: SPI READ BUFFER REGISTER

|       |      |      |      |      |      |      |       |
|-------|------|------|------|------|------|------|-------|
| R-0   | R-0  | R-0  | R-0  | R-0  | R-0  | R-0  | R-0   |
| RXB7  | RXB6 | RXB5 | RXB4 | RXB3 | RXB2 | RXB1 | RXB0  |
| bit 7 |      |      |      |      |      |      | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

- bit 7-0 **RXB<7:0>**: Receiver Buffer bits (read-only)  
If RX buffer is not empty:  
 Contains the top-most byte of RxFIFO, and reading this register will remove the top-most byte RxFIFO and decrease the occupancy of the RxFIFO  
If RX buffer is empty:  
 Reading this register will read as '0', leave the occupancy unchanged, and set the RXRE bit of SPIxSTATUS

## REGISTER 32-12: SPIxTxB: SPI TRANSMIT BUFFER REGISTER

|       |      |      |      |      |      |      |       |
|-------|------|------|------|------|------|------|-------|
| W-0   | W-0  | W-0  | W-0  | W-0  | W-0  | W-0  | W-0   |
| TXB7  | TXB6 | TXB5 | TXB4 | TXB3 | TXB2 | TXB1 | TXB0  |
| bit 7 |      |      |      |      |      |      | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

bit 7-0 **TXB<7:0>**: Transmit Buffer bits (write only)

If TXFIFO is not full:

Writing to this register adds the data to the top of the TXFIFO and increases the occupancy of the TXFIFO write pointer

If TXFIFO is full:

Writing to this register does not affect the data in the TXFIFO or the write pointer, and the TXWE bit of SPIxSTATUS will be set

## REGISTER 32-13: SPIxCLK: SPI CLOCK SELECTION REGISTER

|       |     |     |     |         |         |         |         |
|-------|-----|-----|-----|---------|---------|---------|---------|
| U-0   | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| —     | —   | —   | —   | CLKSEL3 | CLKSEL2 | CLKSEL1 | CLKSEL0 |
| bit 7 |     |     |     |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

bit 7-4 **Unimplemented**: Read as '0'

bit 3-0 **CLKSEL<3:0>**: SPI Clock Source Selection bits

1111-1001 = Reserved  
 1000 = SMT\_match  
 0111 = TMR6\_Postscaled  
 0110 = TMR4\_Postscaled  
 0101 = TMR2\_Postscaled  
 0100 = TMR0\_overflow  
 0011 = CLKREF  
 0010 = MFINTOSC  
 0001 = HFINTOSC  
 0000 = FOSC



**TABLE 32-3: SUMMARY OF REGISTERS ASSOCIATED WITH SPI**

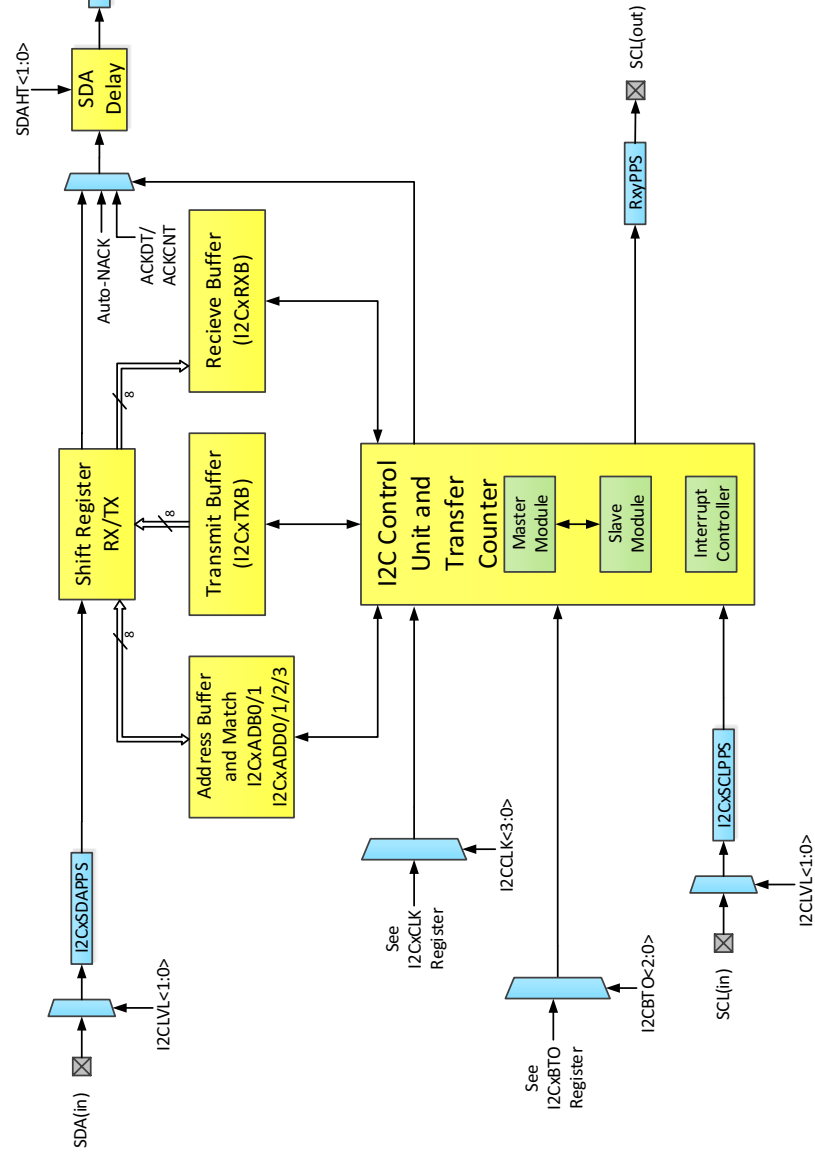
| Name       | Bit 7  | Bit 6 | Bit 5 | Bit 4 | Bit 3   | Bit 2   | Bit 1   | Bit 0   | Register on page    |
|------------|--------|-------|-------|-------|---------|---------|---------|---------|---------------------|
| SPIxINTF   | SRMTIF | TCZIF | SOSIF | EOSIF | —       | RXOIF   | TXUIF   | —       | <a href="#">521</a> |
| SPIxINTE   | SRMTIE | TCZIE | SOSIE | EOSIE | —       | RXOIE   | TXUIE   | —       | <a href="#">522</a> |
| SPIxTCNTH  | —      | —     | —     | —     | —       | TCNT10  | TCNT9   | TCNT8   | <a href="#">523</a> |
| SPIxTCNTL  | TCNT7  | TCNT6 | TCNT5 | TCNT4 | TCNT3   | TCNT2   | TCNT1   | TCNT0   | <a href="#">522</a> |
| SPIxTWIDTH | —      | —     | —     | —     | —       | TWIDTH2 | TWIDTH1 | TWIDH0  | <a href="#">523</a> |
| SPIxBAUD   | BAUD7  | BAUD6 | BAUD5 | BAUD4 | BAUD3   | BAUD2   | BAUD1   | BAUD0   | <a href="#">524</a> |
| SPIxCON0   | EN     | —     | —     | —     | —       | LSBF    | MST     | BMODE   | <a href="#">524</a> |
| SPIxCON1   | SMP    | CKE   | CKP   | FST   | —       | SSP     | SDIP    | SDOP    | <a href="#">525</a> |
| SPIxCON2   | BUSY   | SSFLT | —     | —     | —       | SSET    | TXR     | RXR     | <a href="#">526</a> |
| SPIxSTATUS | TXWE   | —     | TXBE  | —     | RXRE    | CLRBF   | —       | RXBF    | <a href="#">527</a> |
| SPIxRXB    | RXB7   | RXB6  | RXB5  | RXB4  | RXB3    | RXB2    | RXB1    | RXB0    | <a href="#">527</a> |
| SPIxTXB    | TXB7   | TXB6  | TXB5  | TXB4  | TXB3    | TXB2    | TXB1    | TXB0    | <a href="#">528</a> |
| SPIxCLK    | —      | —     | —     | —     | CLKSEL3 | CLKSEL2 | CLKSEL1 | CLKSEL0 | <a href="#">528</a> |

**Legend:** — = unimplemented, read as '0'. Shaded cells are unused by the SPI module.

### 33.0 I<sup>2</sup>C MODULE

The device has two dedicated, independent I<sup>2</sup>C modules. [Figure 33-1](#) is a block diagram of the I<sup>2</sup>C interface module. The figure shows both the Master and Slave modes together.

**FIGURE 33-1: I<sup>2</sup>C MODULE BLOCK DIAGRAM**



## 33.1 I<sup>2</sup>C Features

- Inter-Integrated Circuit (I<sup>2</sup>C) interface supports the following modes in hardware:
  - Master mode
  - Slave mode with byte NACKing
  - Multi-Master mode
- Dedicated Address, Receive and Transmit buffers
- Up to four slave addresses matching
- General Call address matching
- 7-bit and 10-bit addressing with masking
- Start, Restart, Stop, Address, Write, and ACK Interrupts
- Clock Stretching hardware for:
  - RX Buffer Full
  - TX Buffer Empty
  - After Address, Write, and ACK
- Bus Collision Detection with arbitration
- Bus Timeout Detection
- SDA hold time selection
- I<sup>2</sup>C, SMBus 2.0, and SMBus 3.0 input level selections

## 33.2 I<sup>2</sup>C Module Overview

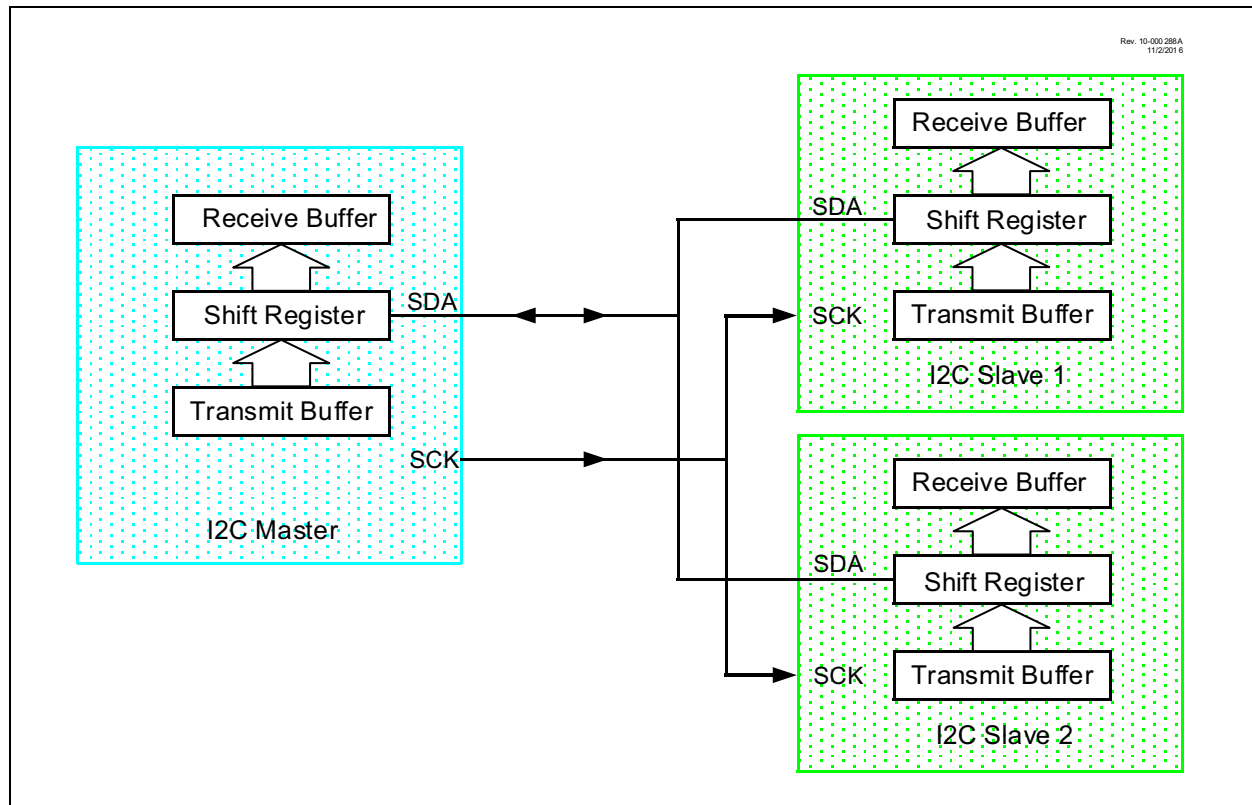
The I<sup>2</sup>C module provides a synchronous interface between the microcontroller and other I<sup>2</sup>C-compatible devices using the two-wire I<sup>2</sup>C serial bus. Devices communicate in a master/slave environment. The I<sup>2</sup>C bus specifies two signal connections:

- Serial Clock (SCL)
- Serial Data (SDA)

Both the SCL and SDA connections are bidirectional open-drain lines, each requiring pull-up resistors to the supply voltage. Pulling the line to ground is considered a logical zero and letting the line float is considered a logical one. Every transaction on the I<sup>2</sup>C bus has to be initiated by the master.

Figure 33-2 shows a typical connection between a master and more than one slave.

**FIGURE 33-2: I<sup>2</sup>C MASTER/SLAVE CONNECTIONS**



There are four main operations based on the direction of the data being shared during I<sup>2</sup>C communication.

- Master Transmit (master is transmitting data to a slave)
- Master Receive (master is receiving data from a slave)
- Slave Transmit (slave is transmitting data to a master)
- Slave Receive (slave is receiving data from the master)

To begin any I<sup>2</sup>C communication, the master device sends out a Start bit followed by the address byte of the slave it intends to communicate with. This is followed by a single Read/Write bit, which determines whether the master intends to transmit to or receive data from the slave device.

If the requested slave exists on the bus, it will respond with an Acknowledge bit, otherwise known as an ACK. The master then continues to shift data in or out of the slave until it terminates the message with a Stop.

Further details about the I<sup>2</sup>C module are discussed in the section below.

## 33.3 I<sup>2</sup>C Mode Operation

All I<sup>2</sup>C communication is 8-bit data and 1-bit acknowledge and shifted out MSb first. The user can control the interaction between the software and the module using several control registers and interrupt flags. Two pins, SDA and SCL, are exercised by the module to communicate with other external I<sup>2</sup>C devices.

### 33.3.1 DEFINITION OF I<sup>2</sup>C TERMINOLOGY

The I<sup>2</sup>C communication protocol terminologies are defined for reference below in [Table 33-1](#). These terminologies are used throughout this document. [Table 33-1](#) has been adapted from the Phillips I<sup>2</sup>C specification.

**TABLE 33-1: I<sup>2</sup>C BUS TERMS**

| TERM             | Description                                                                                                                                                                         |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Transmitter      | The device which shifts data out onto the bus                                                                                                                                       |
| Receiver         | The device which shifts data in from the bus                                                                                                                                        |
| Master           | The device that initiates a transfer, generates clock signals and terminates a transfer                                                                                             |
| Slave            | The device addressed by the master                                                                                                                                                  |
| Multi-master     | A bus with more than one device that can initiate data transfers                                                                                                                    |
| Arbitration      | Procedure to ensure that only one master at a time controls the bus. Winning arbitration ensures that the message is not corrupted                                                  |
| Synchronization  | Procedure to synchronize the clocks of two or more devices on the bus.                                                                                                              |
| Idle             | No master is controlling the bus, and both SDA and SCL lines are high                                                                                                               |
| Active           | Any time one or more master devices are controlling the bus                                                                                                                         |
| Addressed Slave  | Slave device that has received a matching address and is actively being clocked by a master                                                                                         |
| Matching Address | Address byte that is clocked into a slave that matches the value stored in I2CxADR                                                                                                  |
| Write Request    | Slave receives a matching address with R/W bit clear and is ready to clock in data                                                                                                  |
| Read Request     | Master sends an address byte with the R/W bit set, indicating that it wishes to clock data out of the slave. This data is the next and all following bytes until a Restart or Stop. |
| Clock Stretching | When a device on the bus holds SCL low to stall communication                                                                                                                       |
| Bus Collision    | Any time the SDA line is sampled low by the module while it is outputting and expected high state.                                                                                  |
| Bus Timeout      | Any time the I2CBTOISM input transitions high, the I <sup>2</sup> C module is reset and the module goes Idle.                                                                       |

## 33.3.2 BYTE FORMAT

All communication in I<sup>2</sup>C is done in 9-bit segments. A byte is sent from a master to a slave or vice-versa, followed by an Acknowledge bit sent by the receiver. After the 8th falling edge of the SCL line, the device transmitting data on the SDA line releases control of that pin to an input, and reads in an acknowledge value on the next clock pulse. The clock signal is provided by the master. Data is valid to change while the SCL line is low, and sampled on the rising edge of the clock. Changes on the SDA line while the SCL line is high define Start and Stop conditions on the bus which are explained further in the chapter.

## 33.3.3 SDA AND SCL PINS

The user must configure these pins as open-drain outputs. This is done by clearing the appropriate TRIS bits and setting the appropriate ODCON bits. The user may also select the input threshold, slew-rate and internal pull-up settings using the Rxyl2C control registers (Register 16-9).

## 33.3.4 SDA HOLD TIME

The hold time of the SDA pin is selected by the SDAHT<1:0> bits of the I2CxCON2 register. Hold time is the time SDA is held valid after the falling edge of SCL. A longer hold time setting may help on buses with large capacitance.

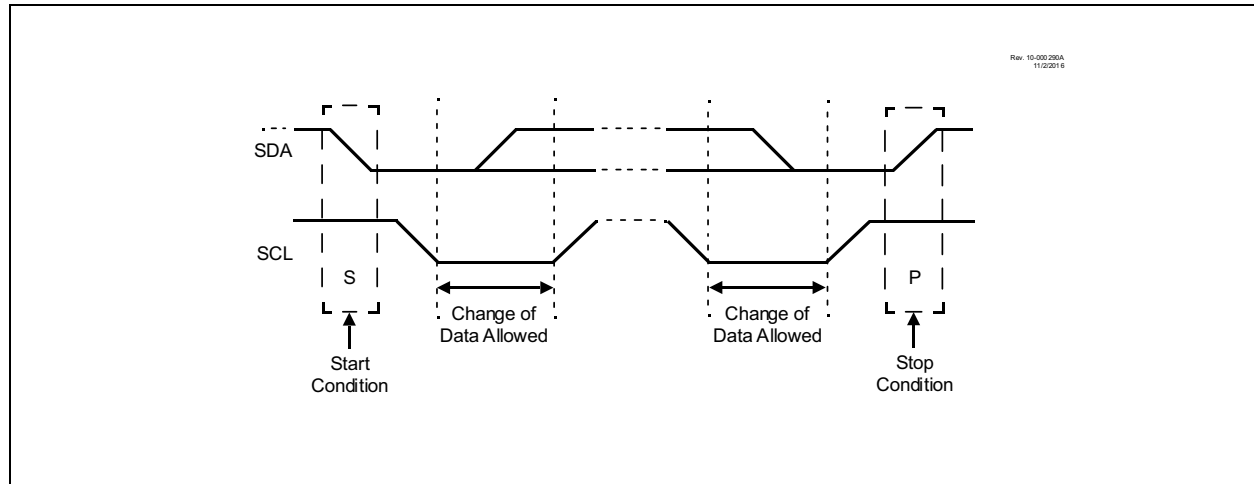
## 33.3.5 START CONDITION

The I<sup>2</sup>C specification defines a Start condition as a transition of SDA line from a high to a low state while SCL line is high. A Start condition is always generated by the master and signifies the transition of the bus from an Idle to an Active state. Figure 33-3 shows waveforms for Start conditions. Master hardware waits for the BFRE bit of I2CxSTAT0 to be set, before asserting a Start condition on the SCL and SDA lines. If two masters assert a start at the same time, a collision will occur during the addressing phase.

## 33.3.6 STOP CONDITION

A Stop condition is a transition of the SDA line from low to high while the SCL line is high. Figure 33-3 shows waveforms for Stop conditions.

**FIGURE 33-3: START AND STOP CONDITIONS**



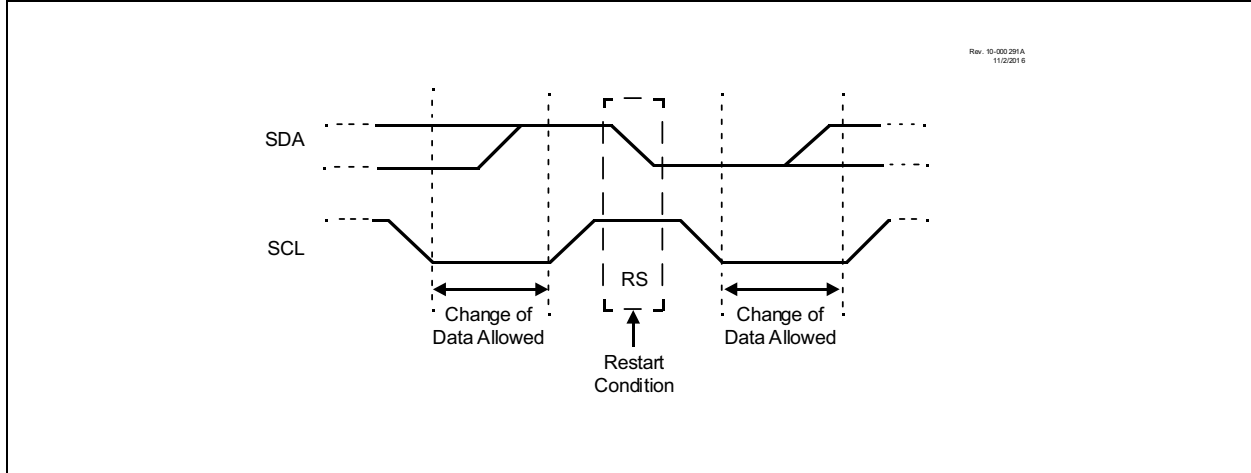
**Note:** At least one SCL low time must appear before a Stop is valid. Therefore, if the SDA line goes low then high again while the SCL line is high, only the Start condition is detected.

## 33.3.7 RESTART CONDITION

A Restart is valid any time that a Stop would be valid. A master can issue a Restart if it wishes to hold the bus after terminating the current transfer. A Restart has the same effect on the slave that a Start would, resetting all slave logic and preparing it to clock in an address. The master may want to address the same or another slave. Figure 33-4 shows the waveform for a Restart condition.

In 10-bit Addressing Slave mode a Restart is required for the master to clock data out of the addressed slave. Once a slave has been fully addressed, matching both high and low address bytes ( $SMA = 1$ ), the master can issue a Restart and the high address byte with the R/W bit set. The slave logic will then hold the clock and prepare to clock out data.

**FIGURE 33-4: RESTART CONDITION**



## 33.3.8 ACKNOWLEDGE SEQUENCE

The ninth SCL pulse for any transferred byte in I<sup>2</sup>C is dedicated as an Acknowledge. It allows receiving devices to respond back to the transmitter by pulling the SDA line low. The transmitter must release control of the line during this time to shift in the response. The Acknowledge (ACK) is an active-low signal, pulling the SDA line low indicates to the transmitter that the device has received the transmitted data and is ready to receive more.

The result of an  $\overline{ACK}$  is placed in the ACKSTAT bit of the I2CxCON1 register. The ACKSTAT bit is cleared when the receiving device sends an Acknowledge and is set when the receiving device does not Acknowledge. A slave sends an Acknowledge when it has recognized its address. When in a mode that is receiving data, the  $\overline{ACK}$  data being sent to the transmitter depends on the value of I2xCNT register. ACKDT is the value sent when  $I2xCNT \neq 0$ . When  $I2xCNT = 0$ , the ACKCNT value is used instead.

In Slave mode, if the ADRIE or WRIE bits are set, clock stretching is initiated when there is an address match or when there is an attempt to write to slave. This allows the user to set the  $\overline{ACK}$  value sent back to the transmitter. The ACKDT bit of the I2CxCON1 register is set/cleared to determine the response. Slave hardware will generate an  $\overline{ACK}$  response if the ADRIE or WRIE bits are clear.

Certain conditions will cause a not-ACK (NACK) to be sent automatically. If any of the RXRE, TXRE, RXO, or TXU bits is set, the hardware response is forced to NACK. All subsequent responses from the device for address matches or data will be a NACK response.

## 33.3.9 BUS TIME-OUT

The I2CxBTO register can be used to select the time-out source for the module. The I<sup>2</sup>C module is reset when the selected bus time out signal goes high. This feature is useful for SMBus and PMBus™ compatibility.

For example, Timer2 can be selected as the bus time-out source and configured to count when the SCL pin is low. If the timer runs over before the SCL pin transitioned high, the timer-out pulse will reset the module.

If the module is configured as a slave and a BTO event occurs when the slave is active (i.e., the SMA bit is set), the module is immediately reset. The SMA and CSTR bits are also cleared, and the BTOIF bit is set.

If a BTO event occurs when the module is configured as a master and is active, (i.e., MMA bit is set), and the module immediately tries to assert a Stop condition and also sets the BTOIF bit. The actual generation of the Stop condition may be delayed if the bus is been clock stretched by some slave device. The MMA bit will be cleared only after the Stop condition is generated.

## 33.3.10 ADDRESS BUFFERS

The I<sup>2</sup>C module has two address buffer registers, I2CxADB0 and I2CxADB1. Depending on the mode, these registers are used as either receive or transmit address buffers. See Table 33-2 for data flow directions in these registers. In Slave modes, these registers are only updated when there is an address match. The ADB bit in the I2CxCON2 register is used to enable/disable the address buffer functionality. When disabled, the address data is sourced from the transmit buffer and is stored in the receive buffer.

**TABLE 33-2: ADDRESS BUFFER DIRECTION AS PER I<sup>2</sup>C MODE**

| Modes                | MODE<2:0> | I2CxADB0 | I2CxADB1 |
|----------------------|-----------|----------|----------|
| Slave (7-bit)        | 000       | RX       | —        |
|                      | 001       | RX       | —        |
| Slave (10-bit)       | 010       | RX       | RX       |
|                      | 011       | RX       | RX       |
| Master (7-bit)       | 100       | —        | TX       |
| Master (10-bit)      | 101       | TX       | TX       |
| Multi-Master (7-bit) | 110       | RX       | TX       |
|                      | 111       | RX       | TX       |

### 33.3.10.1 Slave Mode (7-bit)

In 7-bit Slave mode, I2CxADB0 is loaded with the received matching address and R/W data. The I2CxADB1 register is ignored in this mode.

### 33.3.10.2 Slave Mode (10-bit)

In 10-bit Slave mode, I2CxADB0 is loaded with the lower eight bits of the matching received address. I2CxADB1 is loaded with full eight bits of the high address byte, including the R/W bit.

### 33.3.10.3 Master Mode (7-bit)

The I2CxADB0 register is ignored in this mode. In 7-bit Master mode, the I2CxADB1 register is used to copy address data byte, including the R/W value, to the shift register.

### 33.3.10.4 Master Mode (10-bit)

In 10-bit Master mode, the I2CxADB0 register stores the low address data byte value that will be copied to the shift register after the high address byte is shifted out. The I2CxADB1 register stores the high address byte value that will be copied to the shift register. It is up

to the user to specify all eight of these bits, even though the I<sup>2</sup>C specification defines the upper five bits as a constant.

## 33.3.10.5 Multi-Master Mode (7-bit only)

In Multi-Master mode, the device can be both master and slave depending on the sequence of events on the bus. If being addressed as a slave, the I2CxADB0 register stores the received matching slave address byte. If the device is trying to communicate as a master on the bus, the contents of the I2CxADB1 register are copied to the shift register for addressing a slave device.

## 33.3.11 RECEIVE AND TRANSMIT BUFFER

The receive buffer holds one byte of data while another is shifted into the SDA pin. The user can access the buffer by software (or DMA) through the I2CxRXB register. When new data is loaded into the I2CxRXB register, the receive buffer full Status bit (RXBF) is set and reading the I2CxRXB register clears this bit.

If the user tries to read I2CxRXB when it is empty (i.e., RXBF = 0), receive read error bit (RXRE) is set and a NACK will be generated. The user must clear the error bit to resume normal operation.

The transmit buffer holds one byte of data while another can be shifted out through the SDA pin. The user can access the buffer by software (or DMA) through the I2CxTXB register. When the I2CxTXB does not contain any transmit data, the transmit buffer empty Status bit (TXBE) is set. At this point, the user can load another byte into the buffer.

If the user tries to write I2CxTXB when it is NOT empty (i.e., TXBE = 0), transmit write error flag bit (TXRE) is set and the new data is discarded. When TXRE is set, the user must clear this error condition to resume normal operation.

By setting the CLRBF bit in the I2CxSTAT1 register, the user can clear both receive and transmit buffers. CLRBF will also clear the I2CxRXIF and I2CxTXIF bits.

## 33.3.12 CLOCK STRETCHING

When a slave device has not completed processing data, it can delay the transfer of more data through the process of clock stretching. An addressed slave device may hold the SCL clock line low after receiving or sending a bit, indicating that it is not yet ready to continue. The master will attempt to raise the SCL line in order to transfer the next bit, but will detect that the clock line has not yet been released. Since the SCL connection is open-drain, the slave has the ability to hold the line low until it is ready to continue communicating. Clock stretching allows receivers that cannot keep up with a transmitter to control the flow of incoming data.

Clock stretching can be enabled or disabled by the clearing or setting of CSD (clock stretching disable) bit in the I2CxCON1 register. This bit is valid only in the Multi-Master and Slave modes of operation.

## 33.3.12.1 Clock Stretching for Buffer Operations

If enabled, clock stretching is forced during buffer read/write operations. For example, in Slave mode if RXBF = 1 (receive buffer full), the clock will be stretched after the seventh falling edge of SCL. The SCL line is released only after the user reads data from the receive buffer. This ensures that there is never a receive data overflow. In this situation, if clock stretching is disabled, the RXO bit in I2CxCON1 is set indicating a receive overflow. When set, the module will always respond with a NACK.

Similarly, when TXBE = 1 (transmit buffer empty) and I2CCNT! = 0, the clock is stretched after the 8th falling edge of SCL. The SCL line is released only after the user loads new data into the transmit buffer. This ensures that there is never a transmit underflow. In this situation, if clock stretching is disabled, the TXU bit in I2CxCON1 is set indicating a transmit underflow. When set, the module will always respond with a NACK.

## 33.3.12.2 Clock Stretching for Other Slave Operations

There are three Interrupt and Hold bits that provide clock stretching in Slave mode. These bits can also be used in conjunction with the I2CxIE bit in PIRx register to generate system level interrupts.

- Incoming address match interrupt
  - Clock stretching after an incoming matching address byte is enabled by the Address Interrupt and Hold (ADRIE) bit of the I2CxPIE register. When ADRIE = 1, the CSTR bit is set and the SCL line is stretched following the 8th falling edge of SCL of a received matching address. This allows the user to read the received address from the I2CADB0/1 registers and selectively ACK/NACK based on the received address. Clock stretching from ADRIE is released by software clearing the CSTR bit.
- Data Write Interrupt
  - The data write interrupt and hold enable (WRIE) bit is used to enable clock stretching after a received data byte. When WRIE = 1, the CSTR bit is set, and the SCL line is stretched, following the 8th falling SCL edge for incoming slave data. This bit allows user software to selectively ACK/NACK each received data byte. Clock stretching from WRIE is released by software clearing the CSTR bit.

- Acknowledge status
  - The acknowledge status time interrupt and hold enable (ACKTIE) bit is used to enable clock stretching after the ACK phase of a transmission. This bit enables clock stretching for all address/data transactions; address, write, or read. Following the ACK, the slave hardware will set CSTR. Clock stretching from ACKTIE is released by software clearing the CSTR bit.

## 33.3.13 DATA BYTE COUNT

The I2xCNT register is used to specify the number of bytes in a complete I<sup>2</sup>C packet. The value in this register will decrement every time a data byte is received or transmitted from the I<sup>2</sup>C module. The I2xCNT register will not decrement past zero.

If a byte transfer causes the I2xCNT register to decrement to zero, the Count Interrupt Flag bit (CNTIF) in I2CxPIR is set. This flag bit is set on the 9<sup>th</sup> falling edge of SCL for transmit and receive operations.

The I2xCNT register can be auto-loaded if the ACNT bit in the I2CxCON2 register is set. When ACNT bit is set, the data byte following the address byte is loaded into the I2xCNT register.

**Note 1:** I2xCNT decrements on the eighth (receive) or ninth (transmit) falling edge of SCL; writes during this bit time can corrupt the value.

- 2:** If the block size of the message is greater than 255, the I2xCNT register can be updated mid-message to prevent decrement to zero.

## 33.4 I<sup>2</sup>C Slave Mode

The I<sup>2</sup>C Slave mode operates in one of four modes selected in the Mode bits of I2CxCON0. The modes can be divided into 7- and 10-bit Addressing modes. 10-bit Addressing modes operate the same as 7-bit with some additional overhead for handling the larger addresses.

### 33.4.1 SLAVE ADDRESSING MODES

The I2CxADR/1/2/3 registers contain the Slave mode addresses. The first byte received after a Start or Restart condition is compared against the values stored in these registers. If the byte matches a value, it is loaded into the I2CxADB0/1 registers. If the value does not match, there is no response from the module. The I<sup>2</sup>C module can be configured in the following slave configurations.



## 33.4.1.1 7-bit Addresses Mode

In this mode, the LSb of the received data byte is ignored when determining if there is an address match. All four I2CxADR registers are independently compared to the received address byte.

**Note:** Even though 10-bit addressing calls out only ten bits used in the address comparison, all 15 address bits in I2CxADR0/1 are compared in these modes.

## 33.4.1.2 7-bit Addresses with Masking

In this mode, the value in I2CxADR0 is masked with the value in I2CxADR1 to determine if an address match occurred. A second address and mask are also compared from I2CxADR2/3. When Mode<2:0> = 001 or 111, the I2CxADR1/3 registers serve as the mask value for I2CxADR0/2. All seven bits of the address can be masked

## 33.4.1.3 10-bit Addresses

In this mode, the values stored in I2CxADR0 and I2CxADR1 registers are used to create a 10-bit address. A second 10-bit compare address is formed from I2CxADR2 and I2CxADR3.

## 33.4.1.4 10-bit Address with Masking

In this mode, the I2CxADR0/1 registers are used to form a 10-bit address, and the I2CxADR2/3 registers are used to form a 10-bit mask for that address. When MODE<2:0> = 011, the I2CxADR2/3 registers serve as the mask value for the 10-bit address stored in I2CxADR0/1.

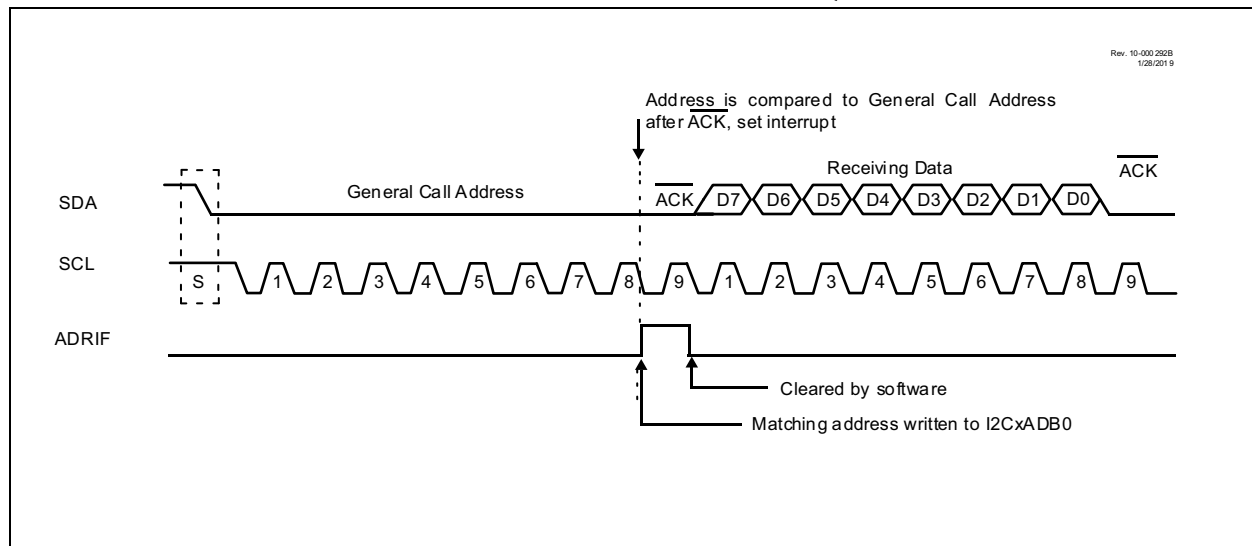
## 33.4.2 GENERAL CALL ADDRESS SUPPORT

The addressing procedure for the I<sup>2</sup>C bus is such that the first byte after the Start condition usually determines which device will be the slave addressed by the master device. The exception is the general call address which can address all devices. When this address is used, all devices should, in theory, respond with an  $\overline{\text{ACK}}$ . The general call address is a reserved address in the I<sup>2</sup>C protocol, defined as address 0x00. In order for the slave hardware to ACK this address, it must be enabled by setting the GCEN bit in the I2CxCON2 register. Setting one of the I2CxADR0/1/2/3 registers to 0x00 is not required. Figure 33-5 shows a General Call reception sequence.

If the ADRIE bit is set, the module will clock stretch after the eighth SCL pulse just like any other address match.

**Note:** General Call addressing is supported in only 7-bit Addressing modes

**FIGURE 33-5: SLAVE MODE GENERAL CALL ADDRESS SEQUENCE**



## 33.4.3 SLAVE OPERATION IN 7-BIT ADDRESSING MODE

The 8th bit in an address byte transmitted by the master is used to determine if the master wants to read from or write to the slave device. If set, it denotes that the master wants to read from the slave and if cleared it means the master wants to write to the slave device. If there is an address match, the R/W bit is copied to the R bit of the I2CxSTAT0 register.

### 33.4.3.1 Slave Reception (7-bit Addressing Mode)

This section describes the sequence of events for the I<sup>2</sup>C module configured as an I<sup>2</sup>C slave in 7-bit Addressing mode and is receiving data. Figure 33-6, Figure 33-7, and Figure 33-8 are used as a visual reference for this description.

1. Master asserts Start condition (can also be a restart) on the bus. Start condition Interrupt Flag (SCIF) in I2CxPIR register is set.
2. If Start condition interrupt is enabled (SCIE bit is set), generic interrupt I2CxIF is set.
3. Master transmits eight bits – 7-bit address and R/W = 0.
4. Received address is compared with the values in I2CxADR0/I2CxADR1/I2CxADR2/I2CxADR3 registers. Refer to section **Section 33.4.1 “Slave Addressing Modes”** for slave addressing modes.
5. If address matches; SMA in I2CxSTAT0 register is set, R/W is copied to R/W bit, D bit is cleared. If the address does not match; module becomes Idle.
6. The matched address data is loaded into I2CxADB0 (if ABD=0) or I2CxRXB (if ABD=1) and ADRIF in I2CxPIR register is set.
7. If Address hold interrupt is enabled (ADRIE = 1), CSTR is set, I2CxIF is set. Slave software can read address from I2CxADB0 and set/clear ACKDT before releasing SCL.
8. If there are any previous error conditions (e.g., Receive buffer overflow or transmit buffer underflow errors), slave will force a NACK and the module becomes Idle.
9. ACKDT value is copied out to SDA for  $\overline{\text{ACK}}$  pulse to be read by the master on the 9th SCL pulse.
10. If the Acknowledge interrupt and hold is enabled (ACKTIE = 1), CSTR is set, I2CxIF is set, then slave software can read address from I2CxADB0 register and change the value of ACKDT before releasing SCL by clearing CSTR.
11. Master sends first seven SCL pulses of the data byte or a Stop condition (in the case of NACK).
12. If Stop condition; PCIF in I2CxPIR register is set, module becomes Idle.
13. If the receive buffer is full from the previous transaction (i.e., RXBF = 1 (I2CxRXIF = 1)), CSTR is set. Slave software must read data out of I2CxRXB to resume communication.
14. Master sends 8th SCL pulse of the data byte. D bit is set, WRIF is set.
15. I2CxRXB is loaded with new data, RXBF bit is set, I2CxRXIF is set.
16. If Data write interrupt and hold is enabled (WRIE = 1), CSTR is set, I2CxIF is set. Slave software can read data from I2CxRXB and set/clear ACKDT before releasing SCL by clearing CSTR.
17. If I2CxCNT = 0, the ACKCNT value is output to the SDA; else, if I2CxCNT! = 0, the ACKDT value is used and the value of I2CxCNT is decremented.
18. The  $\overline{\text{ACK}}$  value is copied out to SDA to be read by the master on the 9th SCL pulse.
19. If I2CxCNT = 0, CNTIF is set.
20. If a NACK was sent, NACKIF is set, module becomes idle.
21. If ACKTIE = 1, CSTR is set, I2CxIF is set. Slave software can read data from I2CxRXB clearing RXBF, before releasing SCL by clearing CSTR.
22. Go to step 11.

**FIGURE 33-6: I<sup>2</sup>C SLAVE, 7-BIT ADDRESS, RECEPTION (ACKTIE = 0, ADRIE = 0, WRIE = 0)**

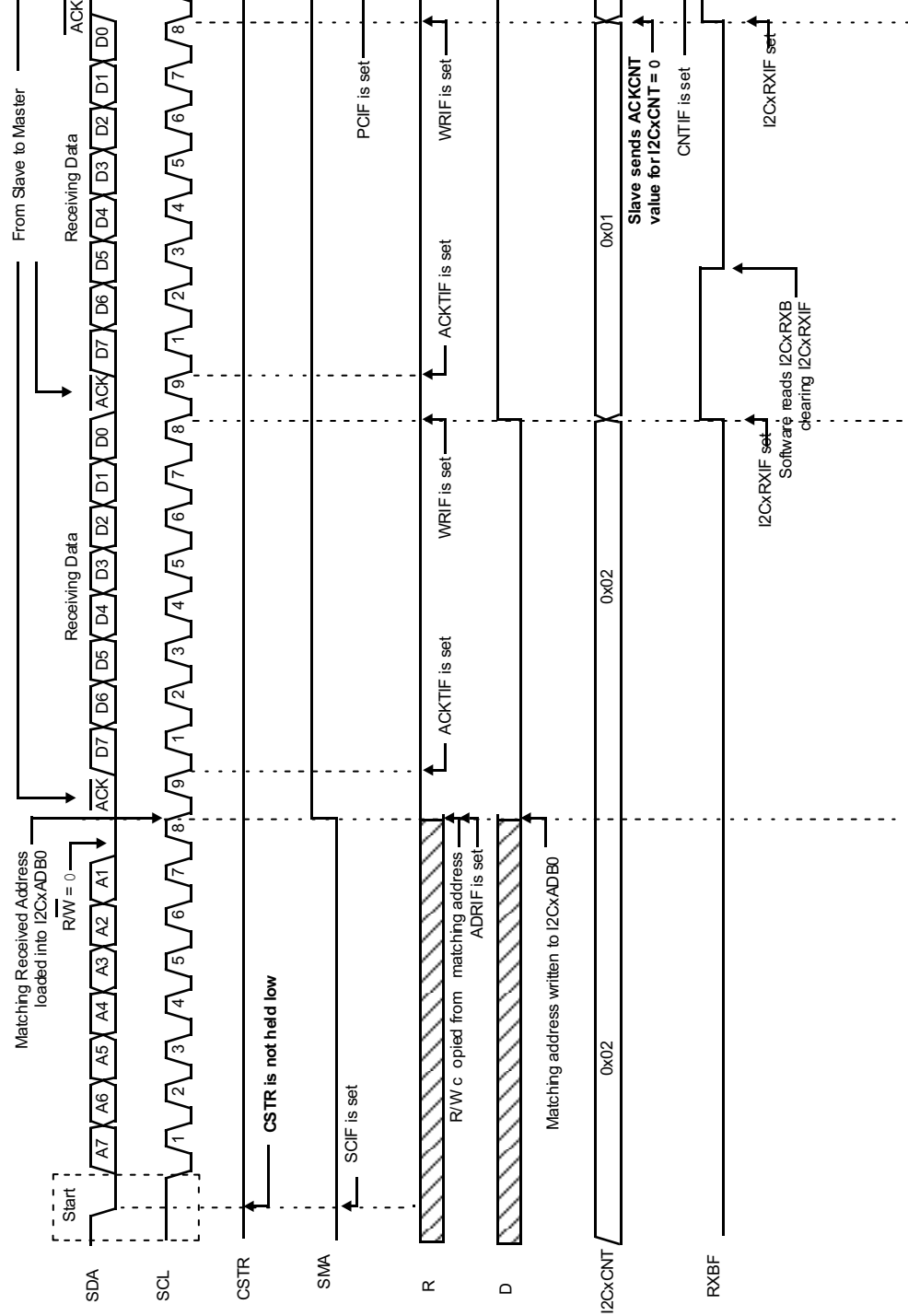


**FIGURE 33-7: I<sup>2</sup>C SLAVE, 7-BIT ADDRESS, RECEPTION WITH I2CxCNT (ACKTIE = 1, ADRIE = 1, ADRIE = 0, WRIE = 0)**

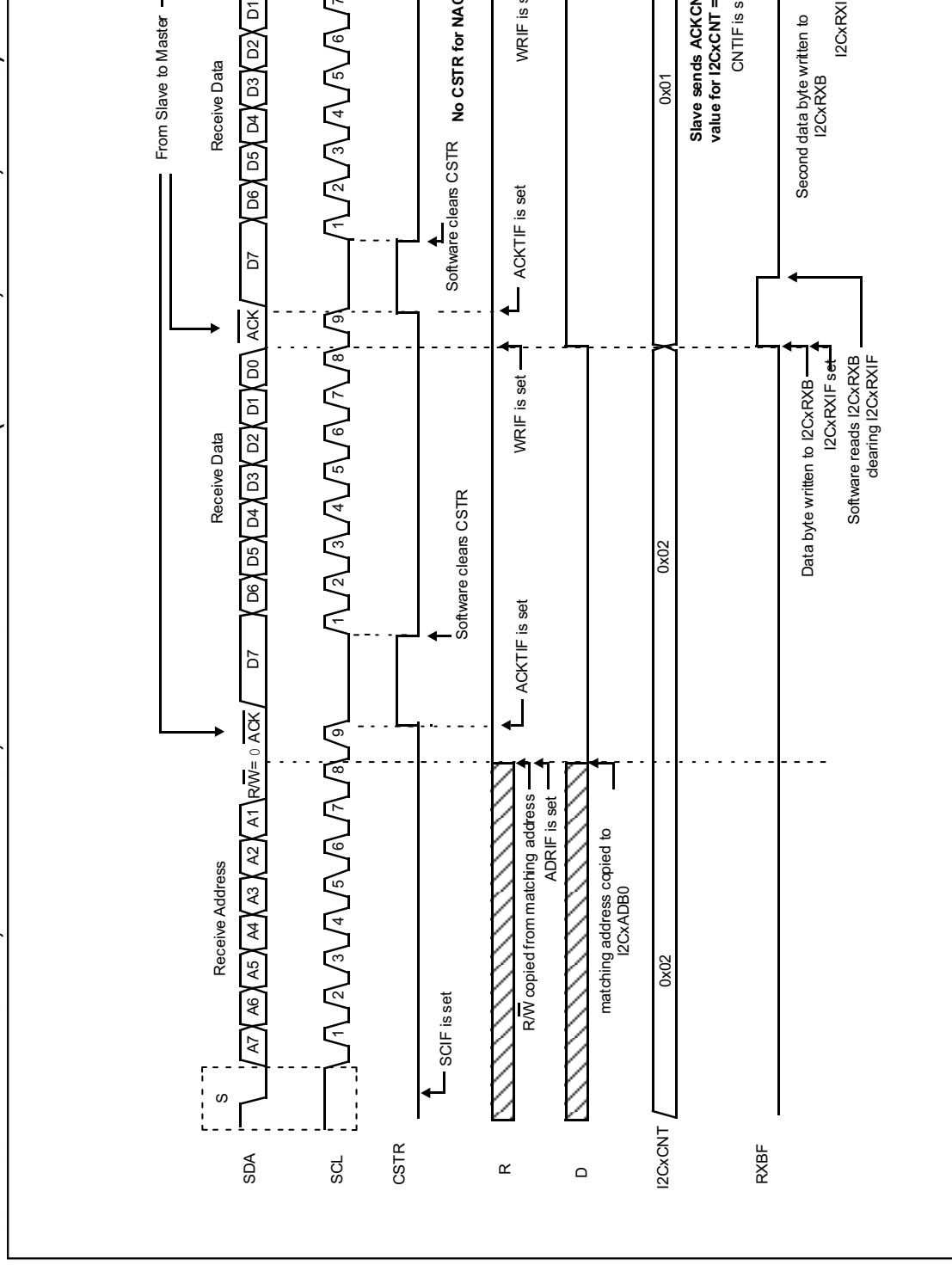
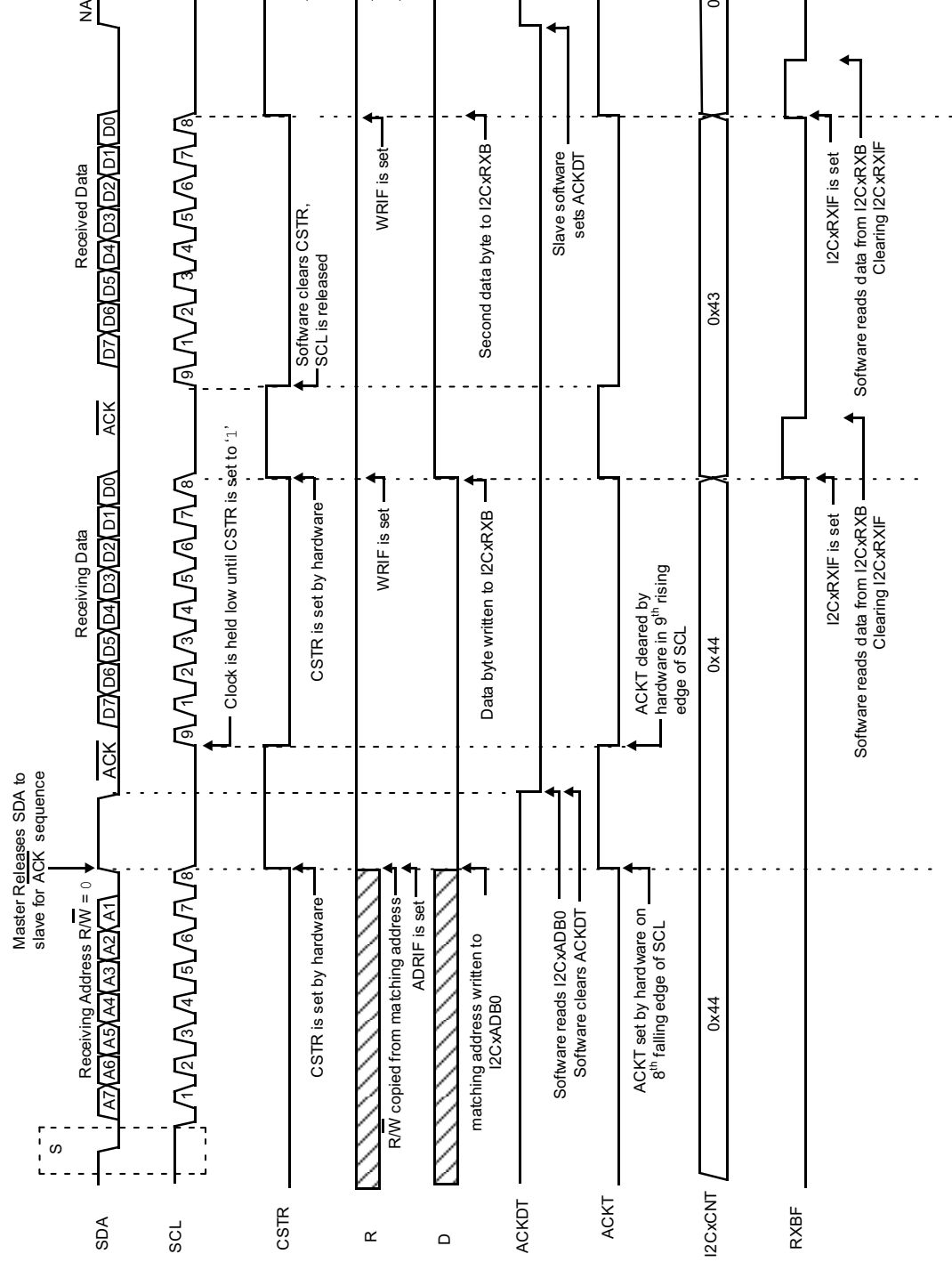


**FIGURE 33-8: I<sup>2</sup>C SLAVE, 7-BIT ADDRESS, RECEPTION (ACKTIE = 0, ADRIE = 1, WRIE = 1)**



## 33.4.3.2 Slave Transmission (7-bit Addressing Mode)

This section describes the sequence of events for the I<sup>2</sup>C module configured as an I<sup>2</sup>C slave in 7-bit Addressing mode and is transmitting data. [Figure 33-9](#) and [Figure 33-10](#) are used as a visual reference for this description.

1. Master asserts Start condition (can also be a restart) on the bus. Start condition Interrupt Flag (SCIF) in I2CxPIR register is set.
2. If Start condition interrupt is enabled (SCIE bit is set), generic interrupt I2CxIF is set.
3. Master transmits eight bits – 7-bit address and R/W = 1.
4. Received address is compared with the values in I2CxADR0/I2CxADR1/I2CxADR2/I2CxADR3 registers. Refer to [Section 33.4.1 “Slave Addressing Modes”](#) for Slave Addressing modes
5. If address matches; SMA in I2CxSTAT0 register is set, R/W is copied to R bit, D bit is cleared. If the address does not match; module becomes idle.
6. The matched address data is loaded into I2CxADB0 (if ABD=0) or I2CxRXB (if ABD=1) and ADRIF in I2CxPIR register is set.
7. If Address hold interrupt is enabled (ADRIE = 1), CSTR is set. I2CxIF is set. Slave software can read address from I2CxADB0 and set/clear ACKDT before releasing SCL. SCL line can be released by clearing CSTR.
8. If the transmit buffer is empty from the previous transaction, i.e., TXBE = 1 and I2CxCNT!= 0 (I2CxTXIF = 1), CSTR is set. Slave software must load data into I2CxTXB to release SCL. I2CxCNT decrements after the byte is loaded into the shift register.
9. Slave hardware waits for 9th SCL pulse with ACK data from master.
10. If I2CxCNT = 0, CNTIF is set.
11. If the Acknowledge interrupt and hold is enabled (ACKTIE = 1), CSTR is set, I2CxIF is set.
12. Slave software can change the value of ACKDT before releasing SCL by clearing CSTR.
13. Master sends eight SCL pulses to clock out data or asserts a Stop condition to end the transaction.
14. Go to step 8.







## 33.4.3.3 Slave operation in 10-bit Addressing Mode

In 10-bit Addressing mode, the first received byte is compared to the binary value of '11110A9A80'. A9 and A8 are the two MSb of the 10-bit address. The first byte is compared with the value in I2CxADR1 and I2CxADR3 registers. After the high byte is acknowledged, the low address byte is clocked in and all eight bits are compared to the low address value in the I2CxADR0 and I2CxADR2 registers. A high and low address match as a write request is required at the start of all 10-bit addressing communication. To initiate a read, the master needs to issue a Restart once the slave is addressed and clock in the high address with the  $\overline{R/W}$  bit set. The slave hardware will then acknowledge the read request and prepare to clock out data. The SMA (slave active) bit is set only when both the high and low address bytes match.

**Note:** All seven bits of the received high address are compared to the values in the I2CxADR1 and I2CxADR3 registers. The five-bit '11110' high address format is not enforced by module hardware. It is up to the user to configure these bits correctly.

## 33.4.3.4 Slave Reception (10-bit Addressing Mode)

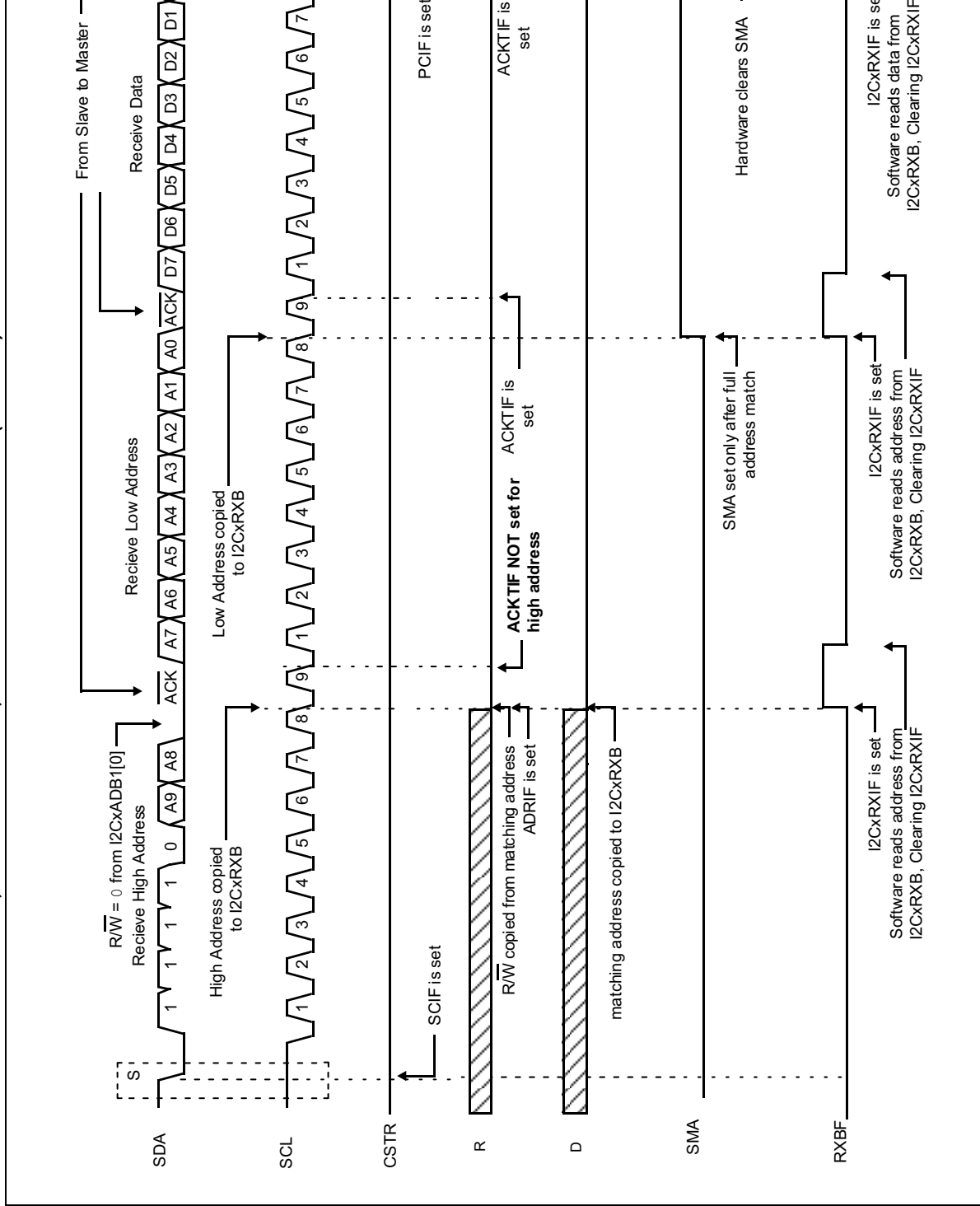
This section describes the sequence of events for the I<sup>2</sup>C module configured as an I<sup>2</sup>C slave in 10-bit Addressing mode and is receiving data. Figure 33-11 is used as a visual reference for this description.

1. Master asserts Start condition (can also be a restart) on the bus. Start condition Interrupt Flag (SCIF) in I2CxPIR register is set. If Start condition interrupt is enabled (SCIE bit is set), generic interrupt I2CxIF is set.
2. Master transmits high address byte with  $\overline{R/W} = 0$ .
3. The received high address is compared with the values in I2CxADR1 and I2CxADR3 registers.
4. If high address matches;  $\overline{R/W}$  is copied to R bit, D bit is cleared, high address data is copied to I2CxADB1. If the address does not match; module becomes idle.
5. If Address hold interrupt is enabled (ADRIE = 1), CSTR is set. I2CxIF is set.
6. Slave software can read high address from I2CxADB1 and set/clear ACKDT before releasing SCL.
7. ACKDT value is copied out to SDA for ACK pulse. SCL line is released by clearing CSTR.
8. Master sends ninth SCL pulse for  $\overline{ACK}$ .
9. Slave can force a NACK at this point due to previous error not being cleared. E.g. Receive buffer overflow or transmit buffer underflow errors. In these cases the slave hardware forces a

NACK and the module becomes Idle.

10. Master transmits low address data byte
11. If the low address matches; SMA is set, ADRIF is set, R/W is copied to R/W bit, D/A bit is cleared, low address data is copied to I2CxADB0, and ACKDT is copied to SDA. If the address does not match; module becomes Idle.
12. If address hold interrupt is enabled, the CSTR bit is set as mentioned in step 6. Slave software can read low address byte from I2CxADB0 register and change ACKDT value before releasing SCL.
13. Master sends ninth SCL pulse for  $\overline{ACK}$ .
14. If the Acknowledge interrupt and hold is enabled (ACKTIE = 1), CSTR is set, I2CxIF is set.
15. Slave software can read address from I2CxADB0 and I2CxADB1 registers and change the value of ACKDT before releasing SCL by clearing CSTR.
16. Master sends first seven SCL pulses of the data byte or a Stop condition (in the case of NACK).
17. If Stop condition; PCIF in I2CxPIR register is set, module becomes Idle.
18. If the receive buffer is full from the previous transaction i.e., RXBF = 1, I2CxRXIF = 1, CSTR is set. Slave software must read data out of I2CxRXB to resume communication.
19. Master sends eighth SCL pulse of the data byte. D bit is set, WRIF is set. I2CxRXB is loaded with new data, RXBF bit is set.
20. If Data write interrupt and hold is enabled (WRIE = 1), CSTR is set, I2CxIF is set. Slave software can read data from I2CxRXB and set/clear ACKDT before releasing SCL by clearing CSTR.
21. If I2CxCNT = 0, the ACKCNT value is output to the SDA; else, the ACKDT value is used and the value of I2CxCNT is decremented.
22. Master sends SCL pulse for  $\overline{ACK}$ .
23. If I2CxCNT = 0, CNTIF is set.
24. If the response was a NACK; NACKIF is set, module becomes idle.
25. If ACKTIE = 1, CSTR is set, I2CxIF is set. Slave software can read data from I2CxRXB clearing RXBF; before releasing SCL by clearing CSTR.
26. Go to step 16.

**FIGURE 33-11: I<sup>2</sup>C SLAVE, 10-BIT ADDRESS, RECEPTION WITH STOP (ADB = 1)**



## 33.4.3.5 Slave Transmission (10-bit Addressing Mode)

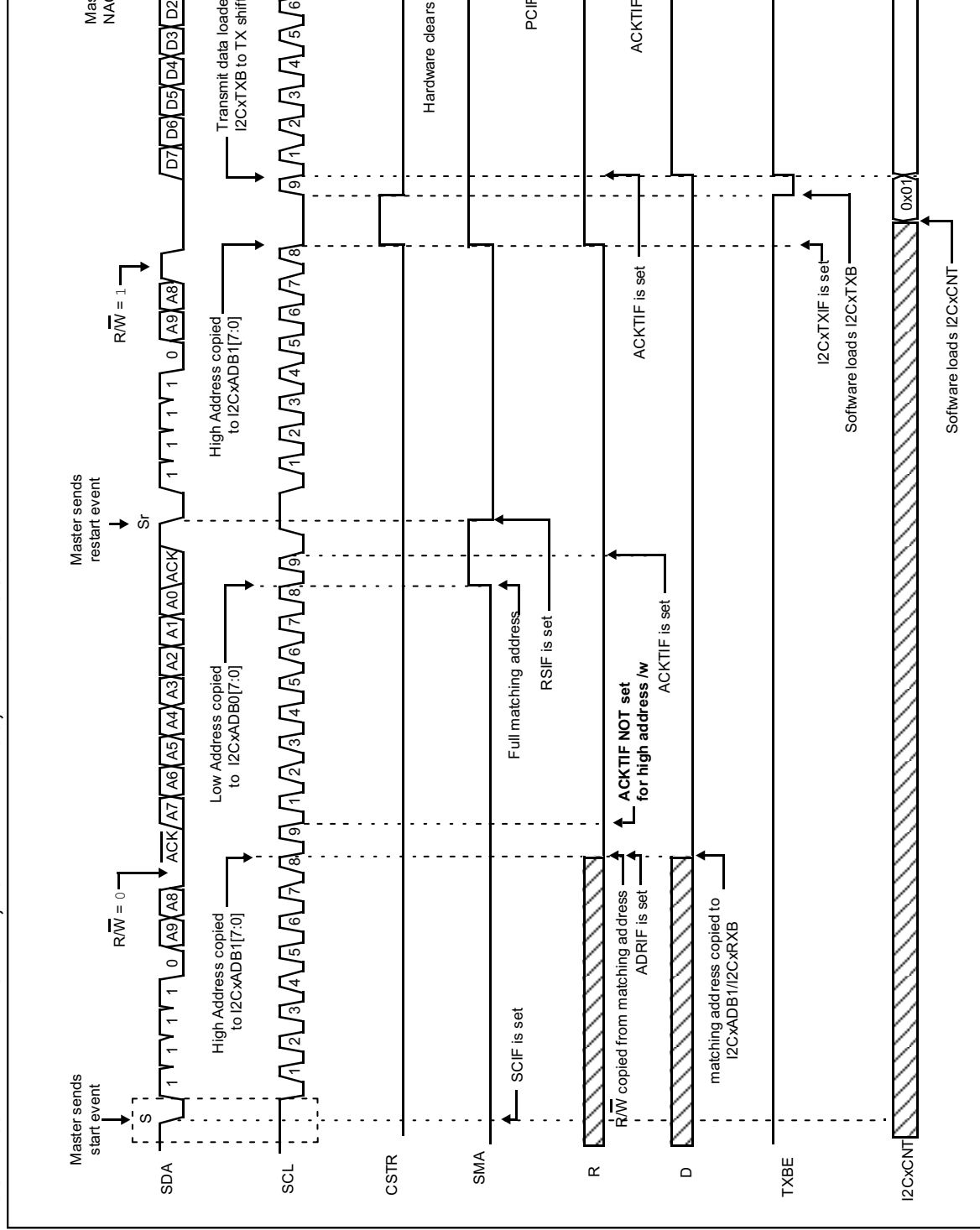
This section describes the sequence of events for the I<sup>2</sup>C module configured as an I<sup>2</sup>C slave in 10-bit Addressing mode and is transmitting data. [Figure 33-12](#) is used as a visual reference for this description.

1. Master asserts Start condition (can also be a restart) on the bus. Start condition Interrupt Flag (SCIF) in I2CxPIR register is set. If Start condition interrupt is enabled (SCIE bit is set), generic interrupt I2CxIF is set.
2. Master transmits high address byte with  $\overline{R/\overline{W}} = 0$ .
3. The received high address is compared with the values in I2CxADR1 and I2CxADR3 registers.
4. If high address matches;  $\overline{R/\overline{W}}$  is copied to R bit, D bit is cleared, high address data is copied to I2CxADB1. If the address does not match; module becomes Idle.
5. If Address hold interrupt is enabled (ADRIE = 1), CSTR is set. I2CxIF is set.
6. Slave software can read high address from I2CxADB1 and set/clear ACKDT before releasing SCL.
7. ACKDT value is copied out to SDA for  $\overline{\text{ACK}}$  pulse. SCL line is released by clearing CSTR.
8. Master sends ninth SCL pulse for ACK.
9. Slave can force a NACK at this point due to previous error not being cleared. E.g. Receive buffer overflow or transmit buffer underflow errors. In these cases the slave hardware forces a NACK and the module becomes Idle.
10. Master transmits low address data byte.
11. If the low address matches; SMA is set, ADRIF is set, low address data is copied to I2CxADB0, and ACKDT is copied to SDA. If the address does not match; module becomes Idle.
12. If address hold interrupt is enabled, the CSTR bit is set as mentioned in step 6. Slave software can read low address byte from I2CxADB0 register and change ACKDT value before releasing SCL.
13. Master sends 9th SCL pulse for  $\overline{\text{ACK}}$ .
14. If the Acknowledge interrupt and hold is enabled (ACKTIE = 1), CSTR is set, I2CxIF is set.
15. Slave software can read address from I2CxADB0 and I2CxADB1 registers and change the value of ACKDT before releasing SCL by clearing CSTR.
16. Master asserts Restart condition (cannot be Start) on the bus. Restart Condition Interrupt Flag (RSCIF) is set. If the Restart Condition Interrupt is enabled, generic interrupt I2CxIF is set.
17. Master transmits high address byte with  $\overline{R/\overline{W}} = 1$ .
18. If SMA = 1, and if high address matches;  $\overline{R/\overline{W}}$  is

copied to R bit, D bit is cleared, high address data is copied to I2CxADB1, and ACKDT is output to SDA. If the address does not match or SMA = 0; module become Idle.

19. If ADRIE = 1, CSTR is set. I2CxIF is set. Slave software can read address from I2CxADB0/1 and set/clear ACKDT. The ACKDT value is copied out to SDA. SCL is released by clearing CSTR bit.
20. If TXBE = 1 and I2CxCNT = 0, I2CxTXIF and CSTR is set. Slave software must load data into I2CxTXB to release SCL.
21. Master sends SCL pulse for  $\overline{\text{ACK}}$ . If I2CxCNT = 0, CNTIF is set.
22. If NACK; NACKIF is set, slave goes Idle.
23. If ACKTIE = 1, CSTR is set, I2CxIF is set. Slave software can read address from I2CxADB0/1 before releasing SCL by clearing CSTR.
24. Master sends eight SCL pulses to clock out data.
25. Go to step 20.

**FIGURE 33-12:**



## 33.5 I<sup>2</sup>C Master Mode

Master mode is enabled by setting and clearing the appropriate MODE<2:0> bits in I2CxCON0 and then by setting the EN bit. Master mode of operation is supported by interrupt generation on buffer full (RXBF), buffer empty (TXBE), and the detection of the Start, Restart, and Stop conditions. The Restart (RS) and Start (S) bits are cleared from a Reset or when the I<sup>2</sup>C module is disabled. Control of the I<sup>2</sup>C bus is asserted when the BFRE bit of I2CSTAT0 is set.

### 33.5.1 I<sup>2</sup>C MASTER MODE OPERATION

The master device generates all of the serial clock pulses and the Start, Restart, and Stop conditions. A transfer is ended with a Stop condition or with a Restart condition. Since the Repeated Start condition is also the beginning of the next serial transfer, the I<sup>2</sup>C bus will not be released, and MMA bit will stay set signifying that the master module is still active.

The steps to initiate a transaction depends on the setting of the address buffer disable bit (ABD) of the I2CxCON2 register.

- ABD = 0 (Address buffers are enabled)

In this case, the master module will use the address stored in the address buffer registers (I2CxADB0/1) to initiate communication with a slave device. User software needs to set the Start bit (S) in the I2CxCON0 register to start communication. This is valid for both 7-bit and 10-bit Addressing modes.

- ABD = 1 (Address buffers are disabled)

In this case, the slave address is transmitted through the transmit buffer and the contents of the address buffers are ignored. User software needs to write the slave address to the transmit buffer (I2CxTXB) to initiate communication. Writing to the Start bit is ignored in this mode. This is valid for both 7-bit and 10-bit Addressing modes.

#### 33.5.1.1 Master Transmitter

In Master Transmitter mode, the first byte transmitted contains the slave address of the receiving device (7 bits) and the Read/Write (R/W) bit. In the case of master transmitter, the R/W bit will be logic '0'. Serial data is transmitted eight bits at a time. After each byte is transmitted, an Acknowledge bit is received. Start and Stop conditions are output to indicate the beginning and the end of a serial transfer.

#### 33.5.1.2 Master Receiver

In Master Receive mode, the first byte transmitted contains the slave address of the transmitting device (7 bits) and the R/W bit. In this case, the R/W bit will be logic '1'. Thus, the first byte transmitted is a 7-bit slave address followed by a '1' to indicate the receive bit. Serial data is received via SDA, while SCL outputs the serial clock. Serial data is received eight bits at a time.

After each byte is received, an Acknowledge bit is transmitted. Start and Stop conditions indicate the beginning and end of the transmission.

### 33.5.2 MASTER CLOCK SOURCE AND ARBITRATION

The I<sup>2</sup>C module clock source is selected by the I2CxCLK register. The I<sup>2</sup>C Clock provides the SCL output clock for Master mode and is used by the Bus Free timer. The I<sup>2</sup>C clock can be sourced from several peripherals.

### 33.5.3 BUS FREE TIME

In Master modes, the BFRE bit of the I2CxSTAT0 register gives an indication of the bus idle status. The master hardware cannot assert a Start condition until this bit is set by the hardware. This prevents the master from colliding with other masters that may already be talking on the bus. The BFRET<1:0> bits of I2CxCON1 allow selection of 8 to 64 pulses of the I<sup>2</sup>C clock input before asserting the BFRE bit. The BFRET bits are used to ensure that the I<sup>2</sup>C module always follows the minimum Stop Hold Time. The I<sup>2</sup>C timing requirements are listed in the electrical specifications chapter.

**Note:** I<sup>2</sup>C clock is not required to have a 50% duty cycle.

### 33.5.4 MASTER CLOCK TIMING

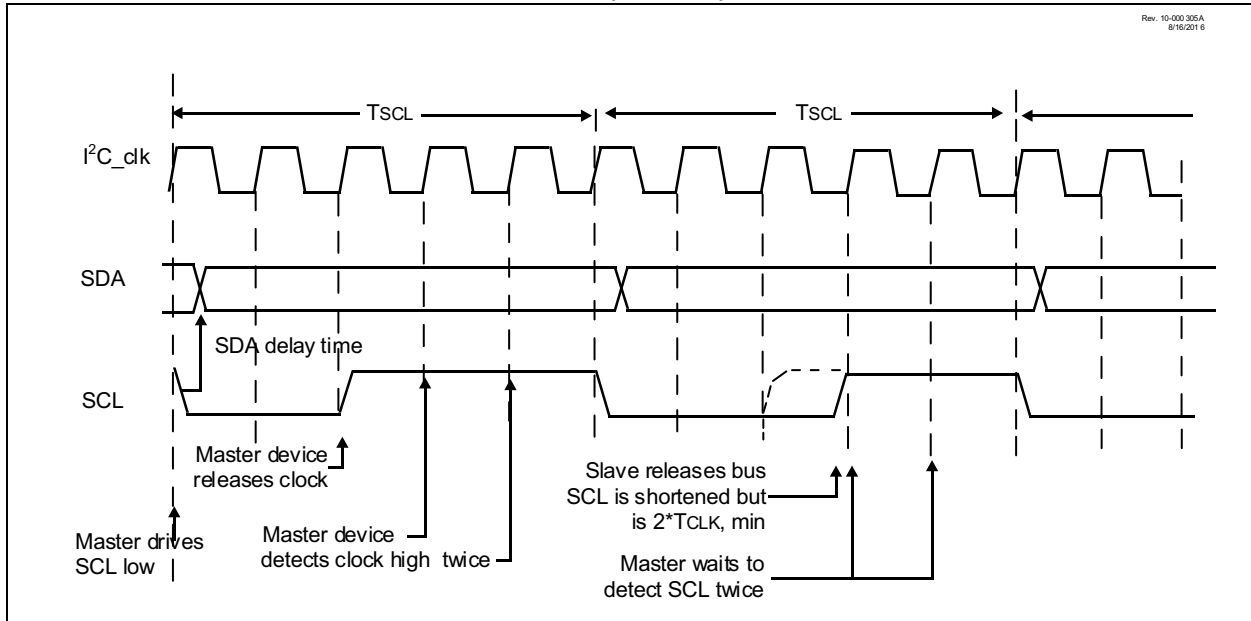
The clock generation in the I<sup>2</sup>C module can be configured using the Fast Mode Enable (FME) bit of the I2CxCON2 register. This bit controls the number of times the SCL pin is sampled before the master hardware drives it.

#### 33.5.4.1 Clock Timing with FME = 0

One TSCL, consists of five clocks of the I<sup>2</sup>C clock input. The first clock is used to drive SCL low, the third releases SCL high. The fourth and fifth clocks are used to detect if the SCL pin is, in fact, high or being stretched by a slave.

If a slave is clock stretching, the hardware waits; checking SCL on each successive I<sup>2</sup>C clock, proceeding only after detecting SCL high. Figure 33-13 shows the clock synthesis timing when FME = 0.

**FIGURE 33-13: CLOCK SYNTHESIS TIMING (FME = 0)**

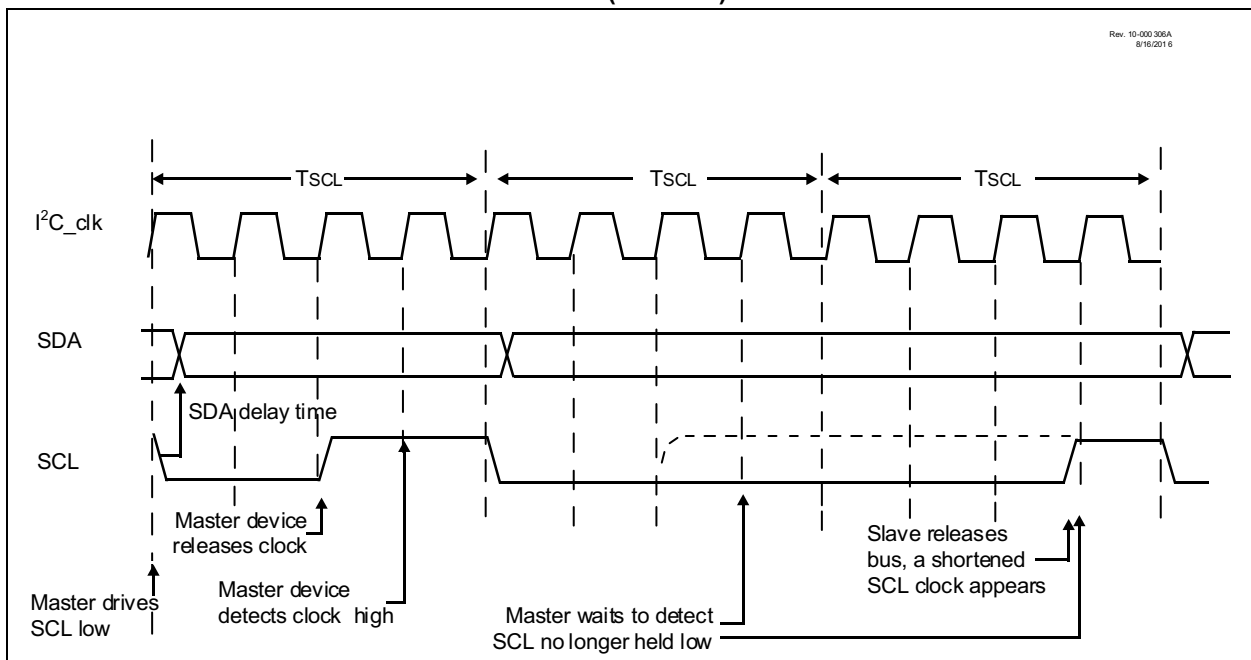


## 33.5.4.2 Clock Timing with FME = 1

One  $T_{SCL}$ , consists of four clocks of the  $I^2C$  clock input. The first clock is used to drive SCL low, the third releases SCL high, and the fourth is used to detect if the clock is, in fact, high or being stretched by a slave.

If a slave is clock stretching, the hardware waits; checking SCL on each successive  $I^2C$  clock, proceeding only after detecting SCL high. Figure 33-14 shows the clock synthesis timing when FME = 1.

**FIGURE 33-14: CLOCK SYNTHESIS TIMING (FME = 1)**

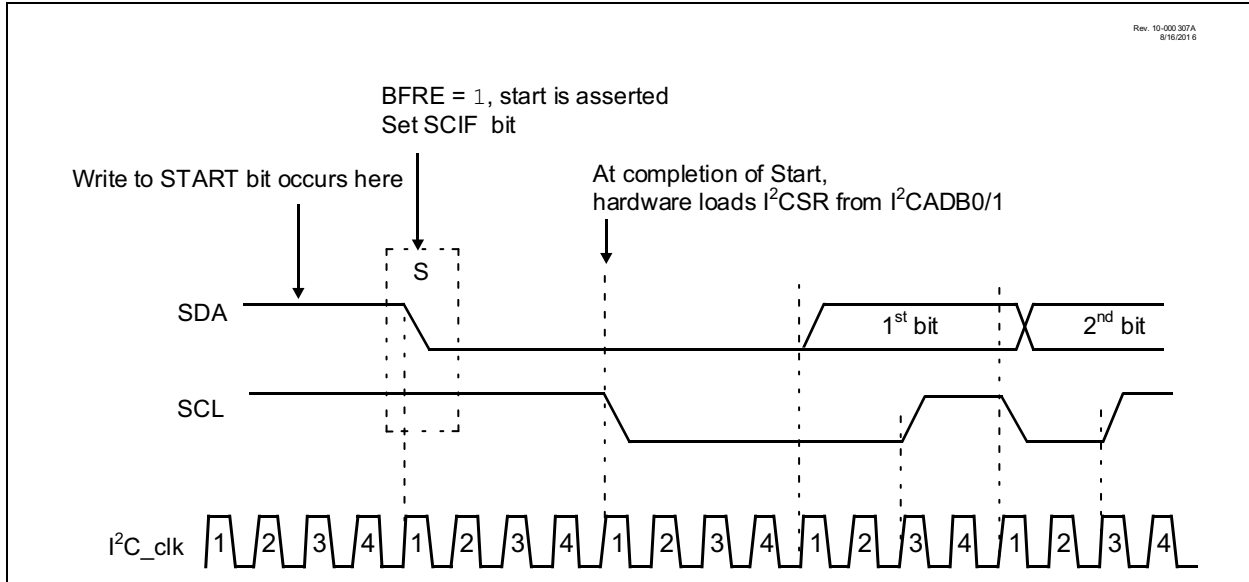


## 33.5.5 I<sup>2</sup>C MASTER MODE START CONDITION TIMING

The user can initiate a Start condition by either writing to the Start bit (S) of the I2CxCON0 register or by writing to the I2CxTXB register based on the ABD bit setting. Master hardware waits for BFRE = 1, before

asserting the Start condition. The action of the SDA being driven low while SCL is high is the Start condition, causing the SCIF bit to be set. One T<sub>SCL</sub> later the SCL is asserted low, ending the start sequence. [Figure 33-15](#) shows the Start condition timing.

**FIGURE 33-15: START CONDITION TIMING**

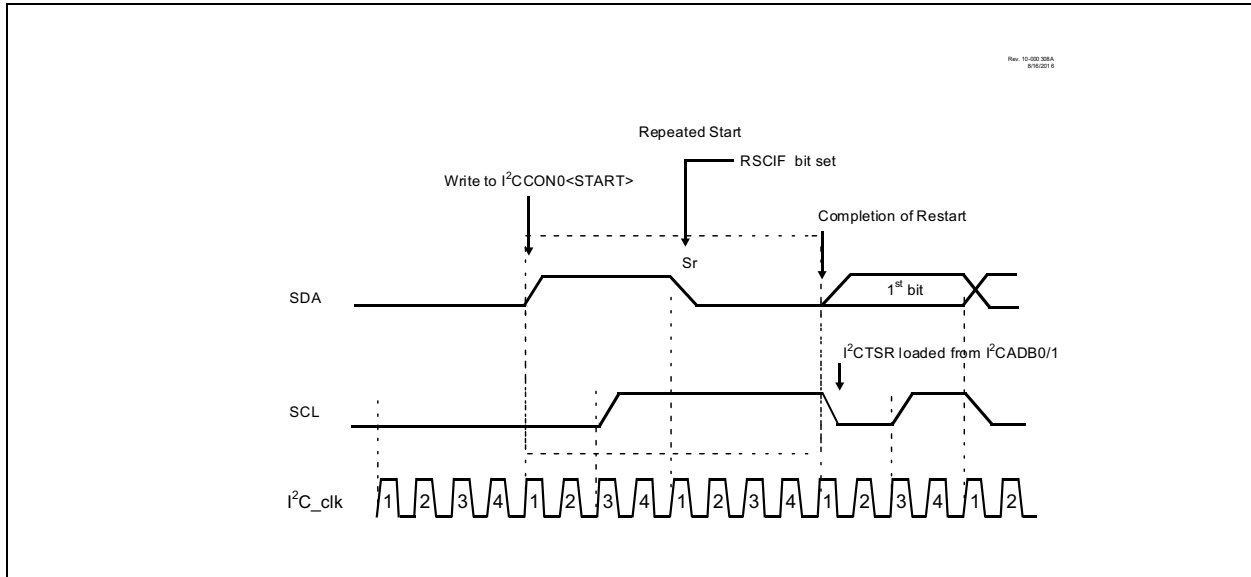


## 33.5.6 I<sup>2</sup>C MASTER MODE REPEATED START CONDITION TIMING

A Repeated Start condition occurs when the Start bit of the I2CxCON0 register is set and the master module is waiting from a Restart clock stretch event (RSEN = 1 and I2CxCNT = 0).

When the Start bit is set, the SDA pin is released high for T<sub>SCL</sub>/2. Then the SCL pin is released floated high) for T<sub>SCL</sub>/2. If the SDA pin is detected low, bus collision flag (BCLIF) is set and the master goes idle. If SDA is detected high, the SDA pin will be pulled low (Start condition) for T<sub>SCL</sub>. Last, SCL is asserted low and I2CxADB0/1 is loaded into the shift register. As soon as a Restart condition is detected on the SDA and SCL pins, the RSCIF bit is set. [Figure 33-16](#) shows the timings for repeated Start Condition.

**FIGURE 33-16: REPEATED START CONDITION TIMING**

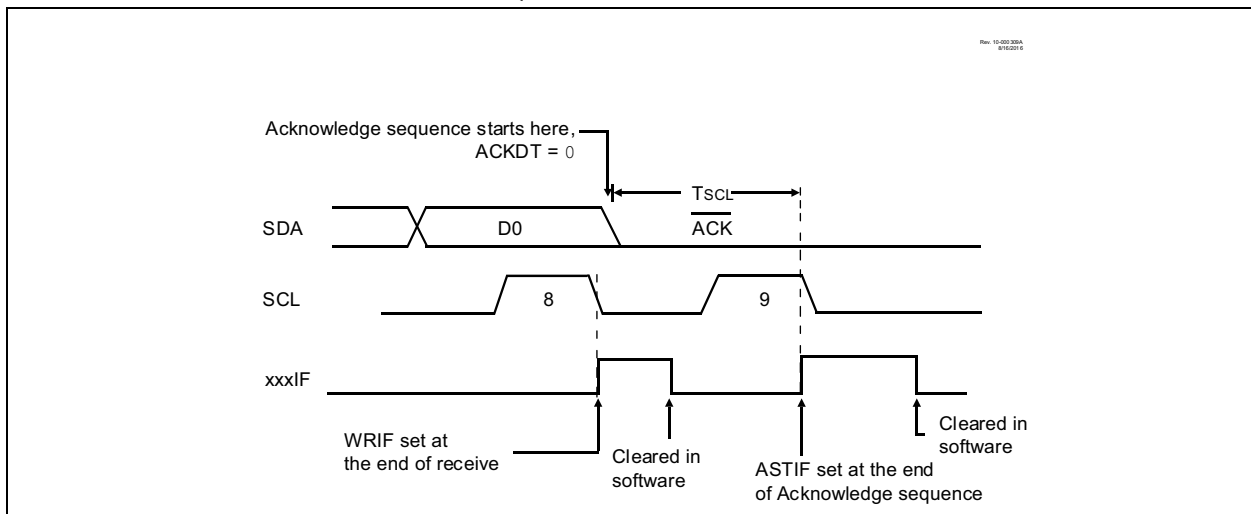


## 33.5.7 ACKNOWLEDGE SEQUENCE TIMING

An Acknowledge sequence is enabled automatically following an address/data byte transmission. The SCL pin is pulled low and the contents of the Acknowledge Data bits (ACKDT/ACKCNT) are presented on the SDA pin. If the user wishes to generate an Acknowledge, then the ACKDT bit should be cleared. If not, the user

should set the ACKDT bit before starting an Acknowledge sequence. The master then waits one clock period (T<sub>SCL</sub>) and the SCL pin is released high. When the SCL pin is sampled high (clock arbitration), the master counts another T<sub>SCL</sub>. The SCL pin is then pulled low. [Figure 33-17](#) shows the timings for Acknowledge sequence.

**FIGURE 33-17: ACKNOWLEDGE SEQUENCE TIMING**



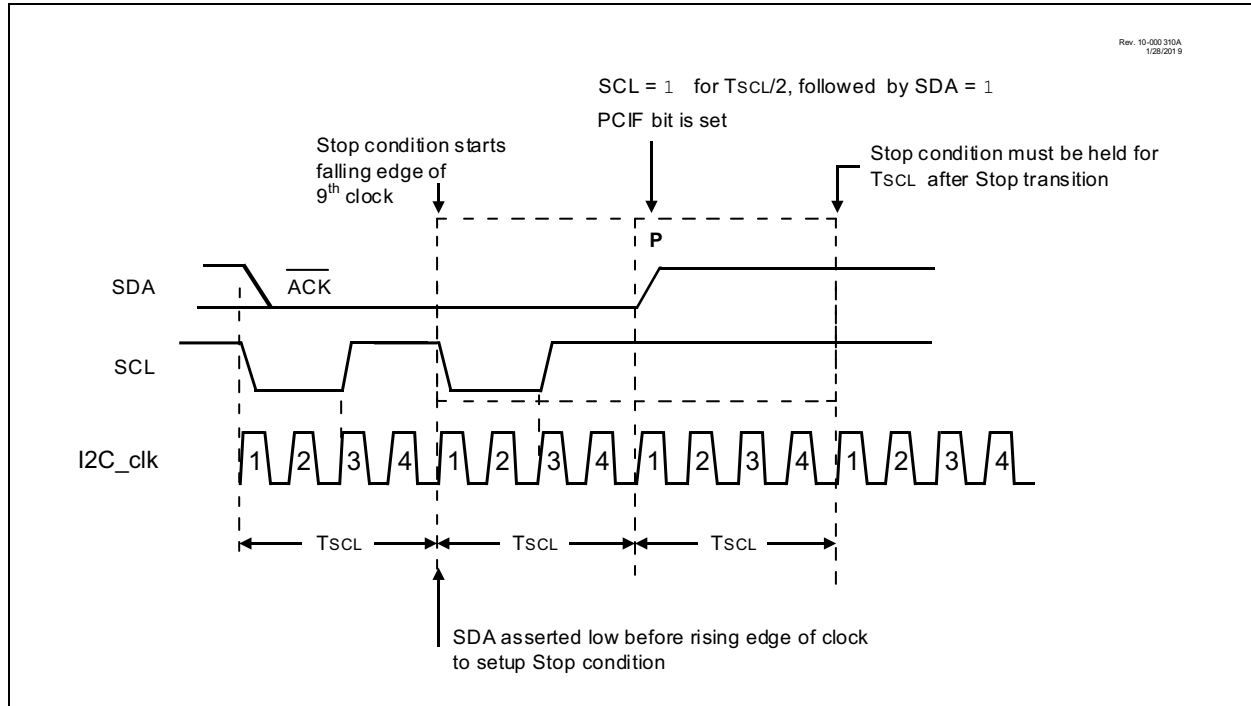
## 33.5.8 STOP CONDITION TIMING

A Stop bit is asserted on the SDA pin at the end of receive/transmit when I2CxCNT = 0. After the last byte of a receive/transmit sequence, the SCL line is held low. The master asserts the SDA line low. The SCL pin is then released high T<sub>SCL</sub>/2 later and is detected high. The SDA pin is then released. When the SDA pin tran-

sitions high while SCL is high, the PCIF bit of the I2CxIF register is set. [Figure 33-18](#) shows the timings for a Stop condition.



**FIGURE 33-18: STOP CONDITION DURING RECEIVE OR TRANSMIT**



## 33.5.9 MASTER TRANSMISSION IN 7-BIT ADDRESSING MODE

This section describes the sequence of events for the I<sup>2</sup>C module configured as an I<sup>2</sup>C master in 7-bit Addressing mode and is transmitting data. [Figure 33-19](#) is used as a visual reference for this description.

1. If ABD = 0; i.e., Address buffers are enabled

Master software loads number of bytes to be transmitted in one sequence in I2CxCNT, slave address in I2CxADB1 with R/W = 0 and the first byte of data in I2CxTXB. Master software has to set the Start (S) bit to initiate communication.

If ABD = 1; i.e., Address buffers are disabled

Master software loads the number of bytes to be transmitted in one sequence in I2CxCNT and the slave address with R/W = 0 into the I2CxTXB register. Writing to the I2CxTXB will assert the start condition on the bus and sets the S bit. Software writes to the S bit are ignored in this case.

2. Master hardware waits for BFRE bit to be set; then shifts out start and address.
3. If the transmit buffer is empty (i.e., TXBE = 1) and I2CxCNT! = 0, the I2CxTXIF and MDR bits are set and the clock is stretched on the 8th falling SCL edge. Clock can be started by loading the next data byte in I2CxTXB register.
4. Master sends out the 9th SCL pulse for ACK.
5. If the master hardware receives ACK from slave device, it loads the next byte from the transmit buffer (I2CxTXB) into the shift register and the

value of I2CxCNT register is decremented.

6. If a NACK was received, master hardware asserts Stop or Restart
7. If ABD = 0; i.e., Address buffers are enabled

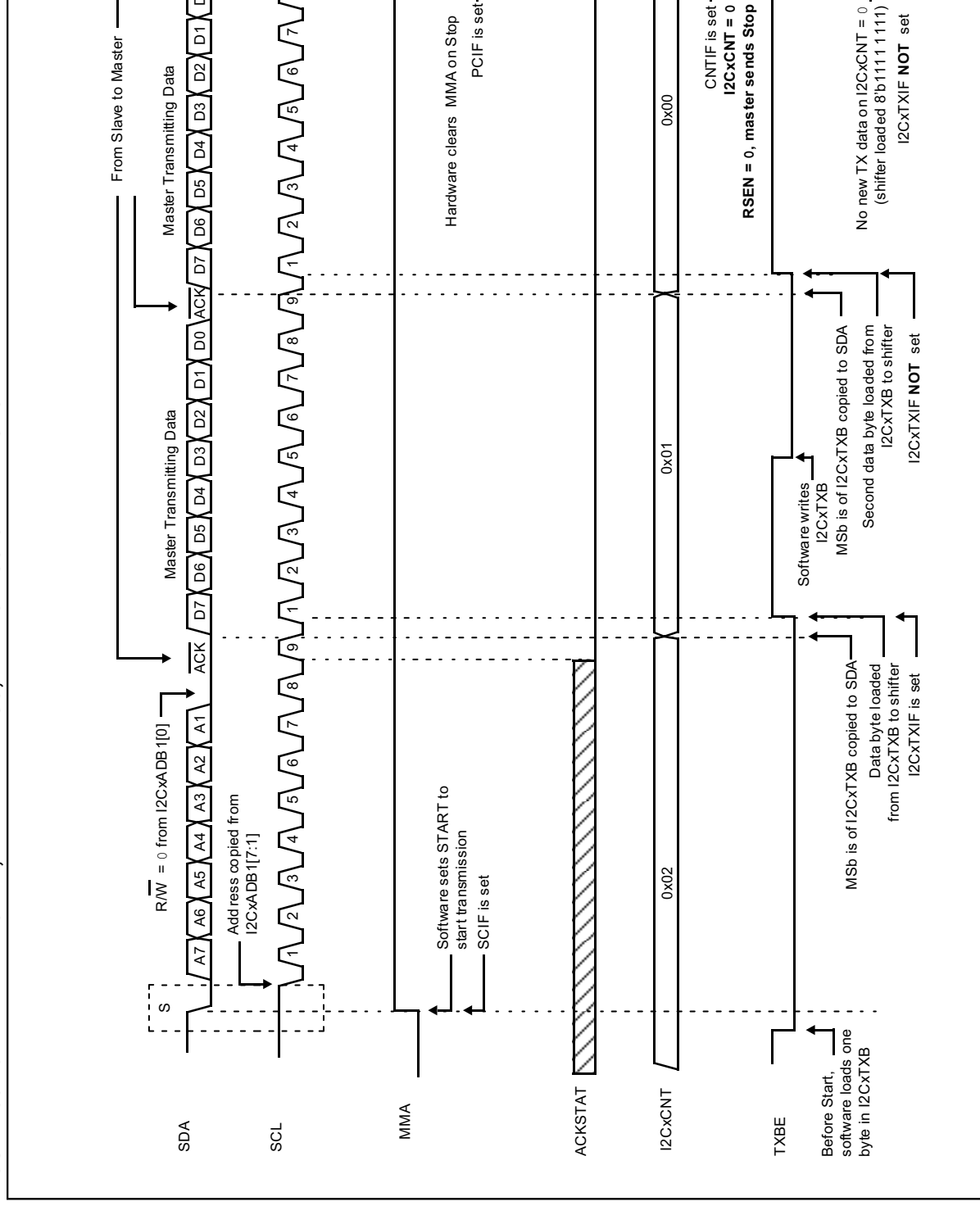
If I2CxCNT = 0, Master hardware sends Stop or sets MDR if RSEN = 1 and waits for the software to set the Start bit again to issue a restart condition.

If ABD = 1; i.e., Address buffers are disabled

If I2CxCNT = 0, Master hardware sends Stop or sets MDR if RSEN = 1 and waits for the software to write the new address to the I2CxTXB register. Software writes to the S bit are ignored in this case.

8. Master hardware outputs data on SDA.
9. If TXBE = 1 and I2CxCNT! = 0, I2CxTXIF and MDR bits are set and the clock is stretched on 8th falling SCL edge. The user can release the clock by writing the next data byte to I2CxTXB register.
10. Master hardware clocks in ACK from slave, and loads the next data byte from I2CxTXB to the shift register. The value of I2CxCNT is decremented.
11. Go to step 7.

**FIGURE 33-19: I<sup>2</sup>C MASTER, 7-BIT ADDRESS, TRANSMISSION WITH STOP**



## 33.5.10 MASTER RECEPTION IN 7-BIT ADDRESSING MODE

This section describes the sequence of events for the I<sup>2</sup>C module configured as an I<sup>2</sup>C master in 7-bit Addressing mode and is receiving data. Figure 33-20 is used as a visual reference for this description.

1. Master software loads slave address in I2CxADB1 with R/W bit = 1 and number of bytes to be received in one sequence in I2CxCNT register.
2. Master hardware waits for BFRE bit to be set; then shifts out start and address with R/W = 1.
3. Master sends out the 9<sup>th</sup> SCL pulse for ACK, master hardware clocks in ACK from slave
4. If ABD = 0; i.e., Address buffers are enabled

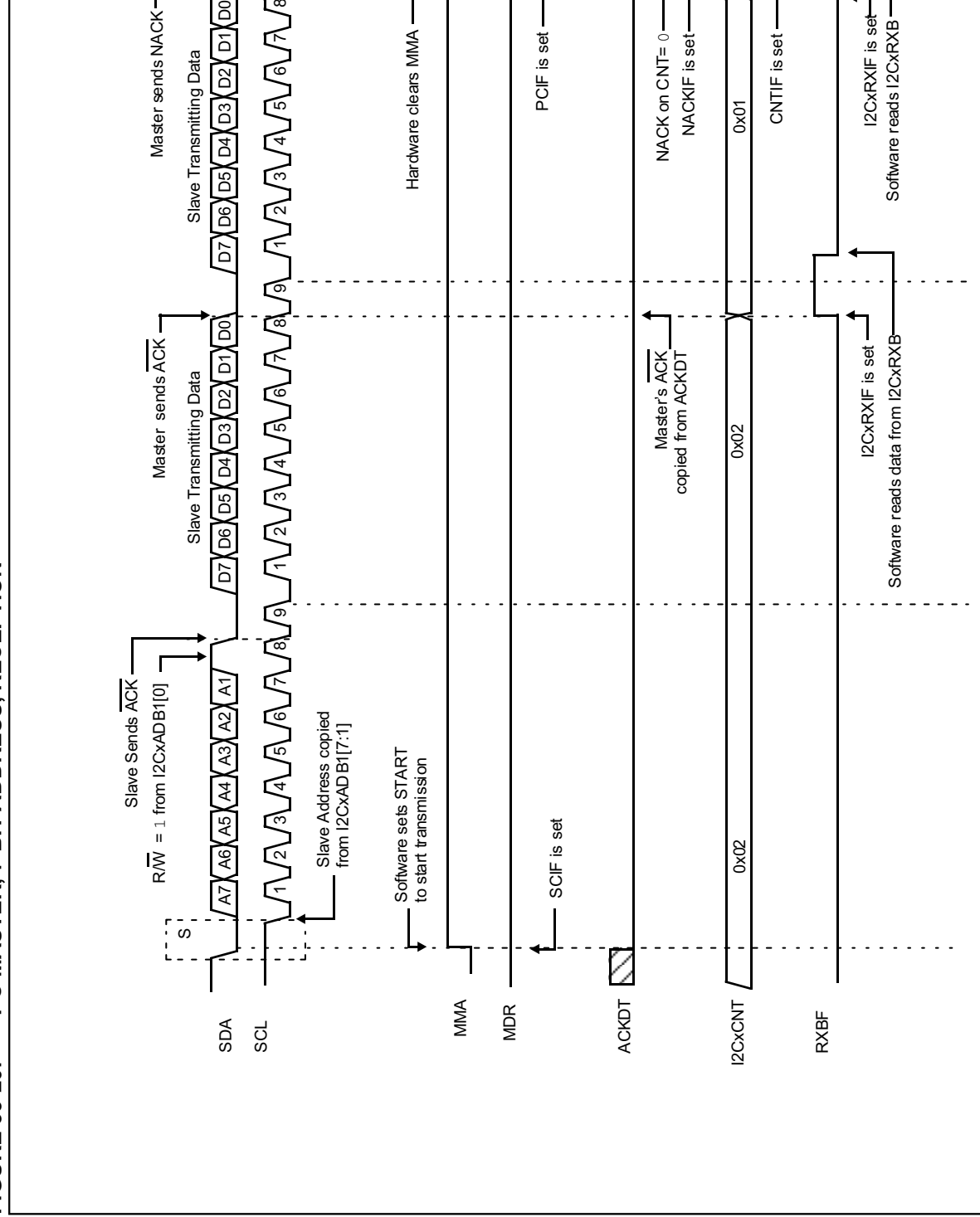
If NACK, master hardware sends Stop or sets MDR (if RSEN = 1) and waits for user software to write to S bit for restart.

If ABD = 1; i.e., Address buffers are disabled

If NACK, master hardware sends Stop or sets MDR (if RSEN = 1) and waits for user software to load the new address into I2CxTXB. Software writes to the S bit are ignored in this case.

5. If ACK, master hardware receives 7 bits of data into the shift register.
6. If the receive buffer is full (i.e., RXBF = 1), clock is stretched on 7<sup>th</sup> falling SCL edge.
7. Master software must read previous data out of I2CxRXB to clear RXBF.
8. Master hardware receives 8th bit of data into the shift register and loads it into I2CxRXB, sets I2CxRXIF and RXBF bits. I2CxCNT is decremented.
9. If I2CxCNT! = 0, master hardware clocks out ACKDT as ACK value to slave. If I2CxCNT = 0, master hardware clocks out ACKCNT as ACK value to slave. It is up to the user to set the values of ACKDT and ACKCNT correctly. If the user does not set ACKCNT to '1', the master hardware will never send a NACK when I2CxCNT becomes zero. Since a NACK was not seen on the bus, the master hardware will also not assert a Stop condition.
10. Go to step 4.

**FIGURE 33-20:**



## 33.5.11 MASTER TRANSMISSION IN 10-BIT ADDRESSING MODE

This section describes the sequence of events for the I<sup>2</sup>C module configured as an I<sup>2</sup>C master in 10-bit Addressing mode and is transmitting data. [Figure 33-21](#) is used as a visual reference for this description

1. If  $ABD = 0$ ; i.e., Address buffers are enabled

Master software loads number of bytes to be transmitted in one sequence in I2CxCNT, high address byte of slave address in I2CxADB1 with  $R/\overline{W} = 0$ , low address byte in I2CxADB0 and the first byte of data in I2CXTXB. Master software has to set the Start (S) bit to initiate communication.

If  $ABD = 1$ ; i.e., Address buffers are disabled

Master software loads the number of bytes to be transmitted in one sequence in I2CxCNT and the high address byte of the slave address with  $R/\overline{W} = 0$  into the I2CXTXB register. Writing to the I2CXTXB will assert the start condition on the bus and sets the S bit. Software writes to the S bit are ignored in this case.

2. Master hardware waits for BFRE bit to be set; then shifts out the start and high address and waits for acknowledge.
3. If NACK, master hardware sends Stop.
4. If  $ABD = 0$ ; i.e., Address buffer are enabled

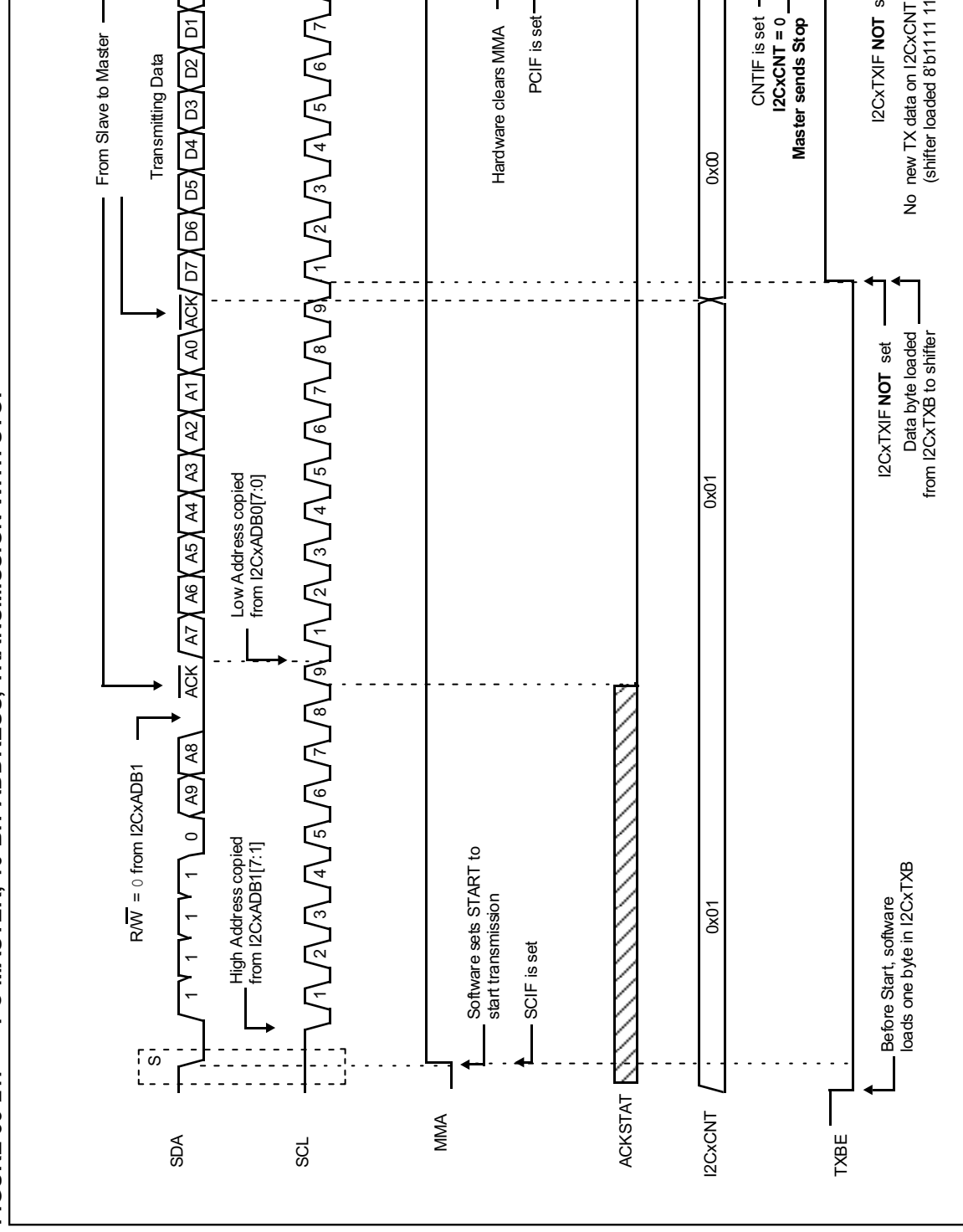
If  $\overline{ACK}$ , master hardware sends the low address byte from I2CxADB0.

If  $ABD = 1$ ; i.e., Address buffer are disabled

If  $\overline{ACK}$ , master hardware sets TXIF and MDR bits and the software has to write the low address byte into I2CXTXB. Writing to I2CXTXB sends the low address on the bus.

5. If  $TXBE = 1$  and  $I2CxCNT \neq 0$ , I2CXTXIF and MDR bits are set. Clock is stretched on 8th falling SCL edge till master software writes next data byte to I2CXTXB.
6. Master hardware sends ninth SCL pulse for  $\overline{ACK}$  from slave and loads the shift register from I2CXTXB. I2CxCNT is decremented.
7. If slave sends a NACK, master hardware sends Stop and ends transmission.
8. If slave sends an  $\overline{ACK}$ , master hardware outputs data in the shift register on SDA. I2CxCNT value is checked on the 8th falling SCL edge. If  $I2CxCNT = 0$ ; master hardware sends 9th SCL pulse for  $\overline{ACK}$  and CNTIF is set.
9. If  $I2CxCNT \neq 0$ ; go to step 5.

**FIGURE 33-21: I<sup>2</sup>C MASTER, 10-BIT ADDRESS, TRANSMISSION WITH STOP**



## 33.5.12 MASTER RECEPTION IN 10-BIT ADDRESSING MODE

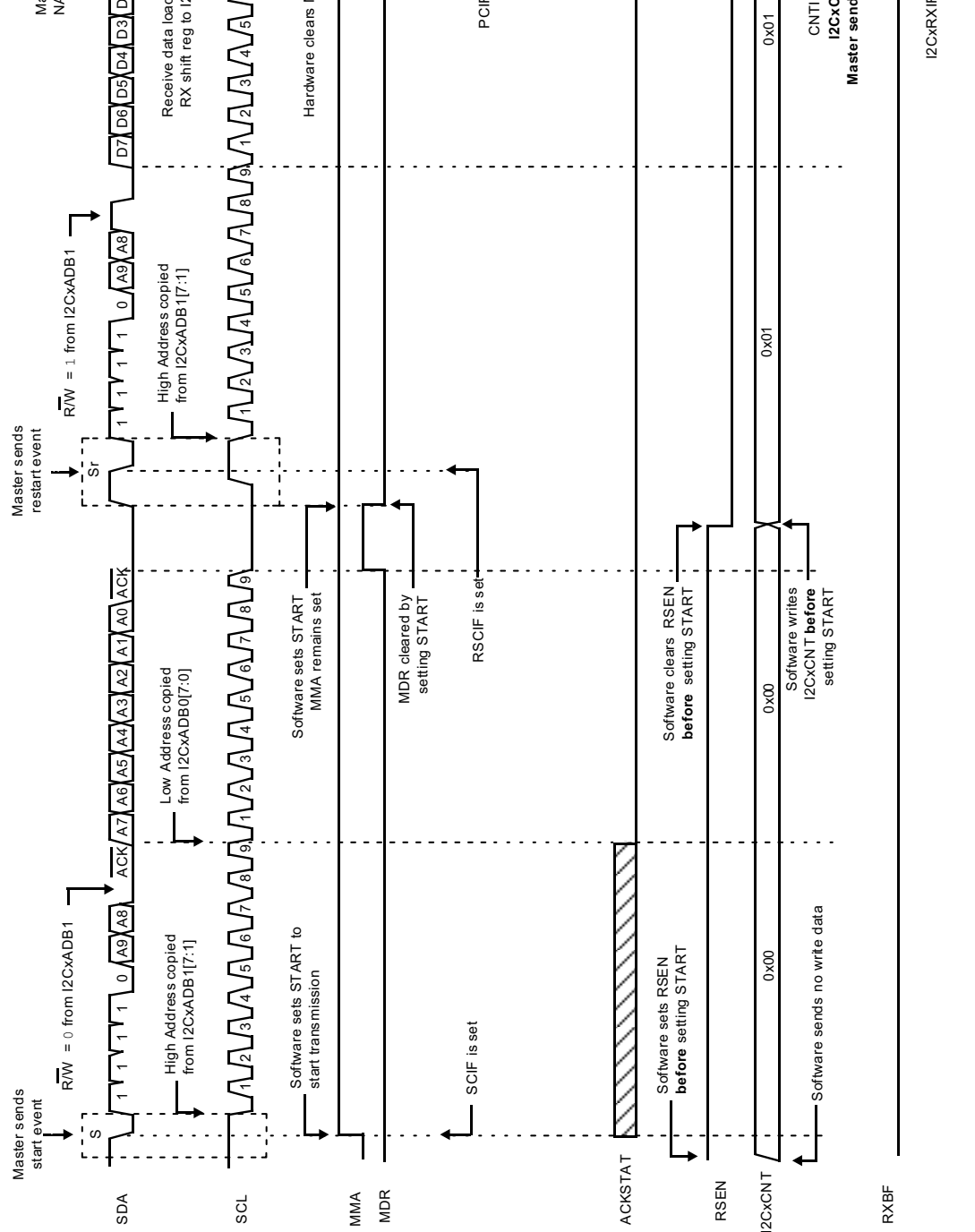
This section describes the sequence of events for the I<sup>2</sup>C module configured as an I<sup>2</sup>C master in 10-bit Addressing mode and is receiving data. Figure 33-22 is used as a visual reference for this description.

1. Depending on the configuration of the Address Buffer Disable (ABD) bit, one of two methods may be used to begin communication:
  - 1.1. When ABD is clear (ABD = 0), the address buffers, I2CxADB0 and I2CxADB1, are enabled. In this case, the address high byte and R/W bit are loaded into I2CxADB1, with R/W clear (R/W = 0). The address low byte is loaded into I2CxADB0, and the Restart Enable (RSEN) bit of I2CxCON0 is set by software. After these registers are loaded, software must set the Start bit to begin communication. Once the S bit is set, master hardware waits for the Bus Free (BFRE) bit to be set before transmitting the Start condition to avoid bus collisions.
  - 1.2. When ABD is set (ABD = 1), the address buffers are disabled. In this case, the number of expected received bytes are loaded into I2xCxNT, the address high byte and R/W bit are loaded into I2CxTXB, with R/W clear (R/W = 0). A write to I2CxTXB will cause master hardware to automatically issue a Start condition once the bus is idle (BFRE = 1). Software writes to the Start bit are ignored.
2. Master hardware waits for BFRE to be set, then shifts out the Start condition. Module hardware sets the Master Mode Active (MMA) bit of I2CxSTAT0 and the Start Condition Interrupt Flag (SCIF) of I2CxPIR. If the Start Condition Interrupt Enable (SCIE) bit of I2CxPIE is also set, the generic I2CxIF is also set.
3. Master hardware transmits the address high byte and R/W bit.
4. Master hardware samples SCL to determine if the slave is stretching the clock, and continues to sample SCL until the line is sampled high.
5. Master hardware transmits the 9th clock pulse, and receives the ACK/NACK response from the slave. If a NACK was received, the NACK Detect Interrupt Flag (NACKIF) is set and the master immediately issues a Stop condition. If an ACK was received, module hardware transmits the address low byte.
6. Master hardware samples SCL to determine if the slave is stretching the clock, and continues to sample SCL until the line is sampled high.
7. Master hardware transmits the 9th clock pulse, and receives the ACK/NACK response from the slave. If an ACK was received, hardware sets MDR, and waits for hardware or software to set the Start bit. If a NACK is received, hardware sets the NACK Detect Interrupt Flag (NACKIF), and:
  - 7.1. ABD = 0: Master generates a Stop condition, or sets the MDR bit (if RSEN is also set) and waits for software to set the Start bit to generate a Restart condition.
  - 7.2. ABD = 1: Master generates a Stop condition, or sets the MDR bit (if RSEN is also set) and waits for software to load a new address into I2CxTXB. Software writes to the Start bit are ignored. If the NACK Detect Interrupt Enable (NACKIE) is also set, hardware sets the generic I2CxEIF bit.
8. Software loads I2CxCNT with the expected number of received bytes.
9. If the ABD is clear (ABD = 0), software sets the Start bit. If the ABD is set (ABD = 1), software writes the address high byte with R/W bit into I2CxTXB, with R/W set (R/W = 1).
10. Master hardware transmits the Restart condition, which sets the Restart Condition Interrupt Flag (RSCIF) bit of I2CxPIR. If the Restart Condition Interrupt Enable (RSCIE) bit of I2CxPIE is also set, the generic I2CxIF is set by hardware.
11. Master hardware transmits the high address byte and R/W bit.
12. Master hardware samples SCL to determine if the slave is stretching the clock, and continues to sample SCL until the line is sampled high.
13. Master hardware transmits the 9th clock pulse, and receives the ACK/NACK response from the slave. If an ACK is received, master hardware receives the first seven bits of the data byte into the receive shift register. If a NACK is received, and:
  - 13.1. ABD = 0: Master generates a Stop condition, or sets the MDR bit (if RSEN is also set) and waits for software to set the Start bit to generate a Restart condition.
  - 13.2. ABD = 1: Master generates a Stop condition, or sets the MDR bit (if RSEN is also set) and waits for software to load a new address into I2CxTXB. Software writes to the Start bit are ignored.
14. If previous data is currently in I2CxRXB (RXBF = 1) when the first seven bits are received by the receive shift register, hardware sets MDR, and the clock is stretched after the 7th falling edge of SCL. This allows software to read I2CxRXB, which clears the RXBF bit, and prevents a receive buffer overflow. Once the RXBF bit is clear, hardware releases SCL.

15. Master hardware clocks in the 8th bit of the data byte into the receive shift register, then transfers the complete byte into I2CxRXB, which sets the I2CxRXIF and RXBF bits. If I2CxRXIE is also set, hardware sets the generic I2CxIF bit. I2CxCNT is decremented by one.
16. Hardware checks I2CxCNT for a zero value. If I2CxCNT is non-zero ( $I2CxCNT \neq 0$ ), hardware transmits the value of the Acknowledge Data (ACKDT) bit as the Acknowledgment response to the slave. It is up to user software to properly configure ACKDT. In most cases, ACKDT should be clear ( $ACKDT = 0$ ), which indicates an ACK response. If I2CxCNT is zero ( $I2CxCNT = 0$ ), hardware transmits the value of the Acknowledge Data (ACKDT) bit as the Acknowledgment response to the slave. CNTIF is set, and master hardware either issues a Stop condition or a Restart condition. It is up to user software to properly configure ACKCNT. In most cases, ACKCNT should be set ( $ACKCNT = 1$ ), which indicates a NACK response. When hardware detects a NACK on the bus, it automatically issues a Stop condition. If a NACK is not detected, the Stop will not be generated, which may lead to a stalled bus condition.
17. Master hardware receives the first seven bits of the next data byte into the receive shift register.
18. Repeat Steps 14-17 until all expected bytes have been received.



**FIGURE 33-22: I<sup>2</sup>C MASTER, 10-BIT ADDRESS, RECEPTION (USING RSEN BIT)**



## 33.6 I<sup>2</sup>C Multi-Master Mode

In Multi-Master mode, the bus-free (BFRE) bit allows the master to determine when the bus is free. Control of the I<sup>2</sup>C bus may be taken when the BFRE bit of the I2CxSTAT0 register is set. Interrupt generation on the detection of a slave address match, ADRIE; causes a clock stretch and allows user software to respond to the master being addressed as a slave device. The Slave Active (SMA) bit is set for a matching received slave address.

Clock arbitration occurs when the master, during any receive, transmit or Restart/Stop condition, releases the SCL pin (SCL allowed to float high). When the SCL pin is allowed to float high, the SCL line is monitored to see if the pin is actually sampled high.

**Note:** In this mode, the slave hardware has priority over the master hardware. Master mode communication can only be initiated when the SMA = 0.

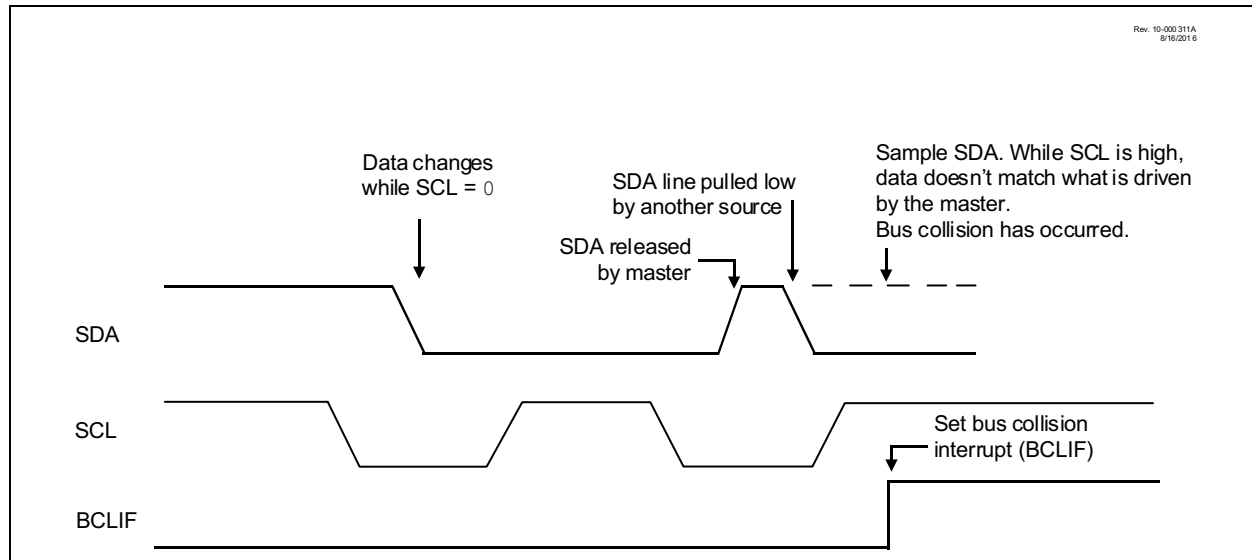
In master operation, the SDA line must be monitored for arbitration to see if the signal level is the expected output level. This check is performed by hardware with the result placed in the BCLIF bit. MMA is cleared when BCLIF is set. The states where arbitration can be lost are:

- Address Transfer
- Data Transfer (master write)
- Repeated Start Condition
- Acknowledge Condition

### 33.6.1 MULTI-MASTER MODE BUS COLLISION

Multi-Master mode support is achieved by bus arbitration. When the master outputs address/data bits onto the SDA pin, arbitration takes place when the master outputs a '1' on SDA, by letting SDA float high and another master asserts a '0'. When the SCL pin floats high, data is stable. If the expected data on SDA is a '1' and the data sampled on the SDA pin is '0', then a bus collision has taken place. The master will set the Bus Collision Interrupt Flag, BCLIF and reset the I<sup>2</sup>C bus to its Idle state. Refer to [Figure 33-23](#) for a detailed timing diagram.

**FIGURE 33-23: BUS COLLISION TIMING FOR TRANSMIT AND ACKNOWLEDGE**



If transmission was in progress when the bus collision occurred, the SDA and SCL lines are released. If a Repeated Start, Stop or Acknowledge was in progress when the bus collision occurred, the action is aborted; the SDA and SCL lines are released. The BCLIF condition must be cleared by software to allow an ACK to be shifted out on the bus again, until then the module will always respond with a NACK. Refer to [Figure 33-24](#) for a detailed timing diagram of a transaction in Multi-Master mode.

DS40001943C-page 563



## 33.7 Register Definitions: I<sup>2</sup>C Control

This section defines all the registers associated with the control and status of the I<sup>2</sup>C bus.

### REGISTER 33-1: I2CxCON0: I<sup>2</sup>C CONTROL REGISTER 0

| R/W-0               | R/W-0 | R/W/HC/HS-0 | R/C/HS/HC-0         | R-0 | R/W-0      | R/W-0 | R/W-0 |
|---------------------|-------|-------------|---------------------|-----|------------|-------|-------|
| EN <sup>(1,2)</sup> | RSEN  | S           | CSTR <sup>(3)</sup> | MDR | MODE <2:0> |       |       |
| bit 7               |       |             |                     |     |            |       | bit 0 |

#### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared | HS = Hardware set HC = Hardware clear                 |

- bit 7 **EN:** I<sup>2</sup>C Module Enable bit  
1 = Enables the I<sup>2</sup>C module<sup>(1,2)</sup>  
0 = Disables the I<sup>2</sup>C module.
- bit 6 **RSEN:** Restart Enable bit (Only MODE<2:0> = 1xx)  
1 = When (I2CxCNT = 0 or ACKSTAT = 1), on 9th falling SCL sets MDR  
0 = When (I2CxCNT = 0 or ACKSTAT = 1), on 9th falling SCL; master shifts out a Stop condition
- bit 5 **S:** Master Start/Restart bit (Only MODE<2:0> = 1xx)  
When MMA = 0  
1 = Set by user set of START bit or write to I2CxTXB, waits for BFRE = 1 to begin with a Start  
0 = Cleared by hardware after sending Start  
When MMA = 1 & MDR = 1  
1 = Set by user set of START bit or write to I2CxTXB, resumes communication with a Restart  
0 = Cleared by hardware after sending Restart  
Else - Writes to I2CxTXB or Start bit (S) has no effect on Start bit
- bit 4 **CSTR:** Slave Clock Stretching bit <sup>(3)</sup>  
1 = Clock is held low (clock stretching)  
0 = Enable clocking, SCL control is released
- SMA = 1 and RXBF = 1 <sup>(6)</sup>  
- Set by hardware on 7th falling SCL edge  
- User must read byte I2CxRXB to release SCL
- SMA = 1 and TXBE = 1 and I2CCNT!= 0  
- Set by hardware on 8th falling SCL edge  
- User must write byte to I2CxTXB to release SCL
- when ADRIE is set <sup>(4)</sup>  
- Set by hardware on 8th falling SCL edge of matching received address  
- User must clear CSTR to release SCL
- SMA = 1 & WRIE = 1  
- Set by hardware on 8th falling SCL edge of received data byte  
- User must clear CSTR to release SCL
- SMA = 1 & ACKTIE = 1  
- Set by hardware on 9th falling SCL edge  
- User must clear CSTR to release SCL

bit 3

**MDR:** Master Data Request (*Master pause*)

- 1 = Master state machine pauses until data is read/written to proceed (SCL is output held low)
- 0 = Master clocking of data is enabled.

MMA = 1 & RXBF = 1

*pause\_for\_rx* - Set by hardware on 7th falling SCL edge  
 - User must read from I2CxRXB to release SCL

MMA = 1 & TXBE = 1 & I2CCNT!= 0

*pause\_for\_tx* - Set by hardware on 8th falling SCL edge  
 - User must write to I2CxTXB to release SCL

*pause\_for\_restart* - Set by hardware on 9th falling SCL edge

RSEN = 1 & MMA = 1 && I2CxCNT = 0 || ACKSTAT = 1

- User must set START or write to I2CxTXB to release SCL and shift Restart onto bus

bit 2-0

**MODE<2:0>:** I<sup>2</sup>C Mode Select bits

- 111 = I<sup>2</sup>C Multi-Master mode (SMBus 2.0 Host), <sup>(5)</sup>  
 Works as both MODE<2:0> = 001 and MODE<2:0> = 100
- 110 = I<sup>2</sup>C Multi-Master mode (SMBus 2.0 Host), <sup>(5)</sup>  
 Works as both MODE<2:0> = 000 and MODE<2:0> = 100
- 101 = I<sup>2</sup>C Master mode, 10-bit address
- 100 = I<sup>2</sup>C Master mode, 7-bit address
- 011 = I<sup>2</sup>C Slave mode, one 10-bit address with masking
- 010 = I<sup>2</sup>C Slave mode, two 10-bit address
- 001 = I<sup>2</sup>C Slave mode, two 7-bit address with masking
- 000 = I<sup>2</sup>C Slave mode, four 7-bit address

**Note 1:** SDA and SCL pins must be configured for open-drain with internal or external pull-up

**2:** SDA and SCL pins must be selected as both input and output in PPS

**3:** CSTR can be set by more than one hardware source, all sources must be addressed by user software before the SCL line is released. CSTR is a module Status bit, and does not show the true bus state.

**4:** SMA is set on the same SCL edge as CSTR for a matching received address

**5:** In this mode, ADRIE should be set, this allows an interrupt to clear the BCLIF condition and allow the  $\overline{\text{ACK}}$  of matching address.

**6:** In 10-bit Slave mode, when ADB = 1, CSTR will set when the high address has not been read out of I2CxRXB before the low address is shifted in.

## REGISTER 33-2: I2CxCON1: I<sup>2</sup>C CONTROL REGISTER 1

| R/W-0                 | R/W-0                  | R-0     | R-0  | U-0 | R/W/HS-0 | R/W/HS-0 | R/W-0 |
|-----------------------|------------------------|---------|------|-----|----------|----------|-------|
| ACKCNT <sup>(2)</sup> | ACKDT <sup>(1,2)</sup> | ACKSTAT | ACKT | —   | RXO      | TXU      | CSD   |
| bit 7                 |                        |         |      |     |          |          | bit 0 |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared | HS = Hardware set HC = Hardware clear                 |

- bit 7 **ACKCNT:** Acknowledge End of Count bit<sup>(2)</sup>  
 Acknowledge value transmitted after received data, when I2CxCNT = 0  
 1 = Not Acknowledge (copied to SDA output)  
 0 = Acknowledge (copied to SDA output)
- bit 6 **ACKDT:** Acknowledge Data bit<sup>(1,2)</sup>  
 Acknowledge value transmitted after matching address  
 Acknowledge value transmitted after received data, when I2CxCNT! = 0  
 1 = Not Acknowledge (copied to SDA output)  
 0 = Acknowledge (copied to SDA output)
- bit 5 **ACKSTAT:** Acknowledge Status bit (Transmission only)  
 1 = Acknowledge was not received for most recent transmission  
 0 = Acknowledge was received for most recent transmission
- bit 4 **ACKT:** Acknowledge Time Status bit  
 1 = Indicates the I<sup>2</sup>C bus is in an Acknowledge sequence, set on 8th falling edge of SCL clock  
 0 = Not in Acknowledge sequence, cleared on 9th rising edge of SCL
- bit 3 **Unimplemented:** Read as 1'b0
- bit 2 **RXO:** Receive Overflow Status bit (MODE<2:0> = 0xx & 11x)  
 This bit can only be set when CSD = 1  
 1 = Set when SMA = 1, and a master clocks in data when RXBF = 1  
 0 = No slave overflow condition
- bit 1 **TXU:** Transmit Underflow Status bit (MODE<2:0> = 0xx & 11x)  
 This bit can only be set when CSD = 1  
 1 = Set when SMA = 1, and a master clocks out data when TXBE = 1  
 0 = No slave underflow condition
- bit 0 **CSD:** Clock Stretching Disable bit (MODE<2:0> = 0xx & 11x)  
 1 = When SMA = 1, the CSTR bit will never be set  
 0 = Slave clock stretching proceeds normally
- Note 1:** Software writes to ACKDT bit must be followed by a minimum SDA data-setup time before clearing CSTR.  
**2:** NACK may still be generated by I<sup>2</sup>C hardware when bus errors are indicated in the I2CxSTAT1 or I2CxERR registers.

# PIC18(L)F25/26K83

## REGISTER 33-3: I2CxCON2: I<sup>2</sup>C CONTROL REGISTER 2

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0      | R/W-0      | R/W-0 | R/W-0 |
|-------|-------|-------|-------|------------|------------|-------|-------|
| ACNT  | GCEN  | FME   | ADB   | SDAHT<1:0> | BFRET<1:0> |       |       |
| bit 7 |       |       |       |            |            |       | bit 0 |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared | HS = Hardware set HC = Hardware clear                 |

- bit 7 **ACNT:** Auto-Load I<sup>2</sup>C Count Register Enable bit  
1 = The first received or transmitted byte after the address, is automatically loaded into the I2CxCNT register. The I2CxCNT register is loaded at the same time as the value is moved to/from the shifter. ACKDT is used to determine the ACK/NACK value for the address bytes and first data byte of a received message. This prevents a NACK from being sent for the byte that would update the I2CxCNT register.  
0 = Auto-load of I2CxCNT disabled
- bit 6 **GCEN:** General Call Address Enable bit (MODE<2:0> = 00x & 11x)  
1 = General call address, 0x00, causes address match event  
0 = General call address disabled
- bit 5 **FME:** Fast Mode Enable bit  
1 = SCL is sampled high only once before driving SCL low. (FSCL = FI2CxCLK/4)  
0 = SCL is sampled high twice before driving SCL low. (FSCL = FI2CxCLK/5)
- bit 4 **ADB:** Address Data Buffer Disable bit  
1 = Received address data is loaded into I2CxRXB  
Transmitted address data is loaded from the I2CxTXB  
0 = Received address data is loaded only into the I2CxADB  
Transmitted address data is loaded from the I2CxADB0/1 registers.
- bit 3-2 **SDAHT<1:0>:** SDA Hold Time Selection bits  
11 = Reserved  
10 = Minimum of 30 ns hold time on SDA after the falling edge of SCL  
01 = Minimum of 100 ns hold time on SDA after the falling edge of SCL  
00 = Minimum of 300 ns hold time on SDA after the falling edge of SCL
- bit 1-0 **BFRET<1:0>:** Bus Free Time Selection bits  
11 = 64 I<sup>2</sup>C Clock pulses  
10 = 32 I<sup>2</sup>C Clock pulses  
01 = 16 I<sup>2</sup>C Clock pulses  
00 = 8 I<sup>2</sup>C Clock pulses

## REGISTER 33-4: I2CxCLK: I<sup>2</sup>C CLOCK SELECTION REGISTER

|       |     |     |     |          |         |         |         |
|-------|-----|-----|-----|----------|---------|---------|---------|
| U-0   | U-0 | U-0 | U-0 | R/W-0/0  | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| —     | —   | —   | —   | CLK<3:0> |         |         |         |
| bit 7 |     |     |     | bit 0    |         |         |         |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

HS = Hardware set

HC = Hardware clear

bit 7-4

**Unimplemented:** Read as '0'

bit 3-0

**CLK<3:0>:** I<sup>2</sup>C Clock Selection Bits

| CLK<3:0>  | I <sup>2</sup> Cx Clock Selection |
|-----------|-----------------------------------|
| 1010-1111 | Reserved                          |
| 1001      | SMT1 overflow                     |
| 1000      | TMR6 post scaled output           |
| 0111      | TMR4 post scaled output           |
| 0110      | TMR2 post scaled output           |
| 0101      | TMR0 overflow                     |
| 0100      | Clock Reference output            |
| 0011      | MFINTOSC (500 kHz)                |
| 0010      | HFINTOSC                          |
| 0001      | Fosc                              |
| 0000      | Fosc/4                            |



## REGISTER 33-5: I2Cx BTO: I<sup>2</sup>C BUS TIMEOUT SELECTION REGISTER

|       |     |     |     |     |          |         |         |
|-------|-----|-----|-----|-----|----------|---------|---------|
| U-0   | U-0 | U-0 | U-0 | U-0 | R/W-0/0  | R/W-0/0 | R/W-0/0 |
| —     | —   | —   | —   | —   | BTO<2:0> |         |         |
| bit 7 |     |     |     |     |          |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

HS = Hardware set

HC = Hardware clear

bit 7-3

**Unimplemented:** Read as '0'

bit 2-0

**BTO<2:0>:** I<sup>2</sup>C Bus Timeout Selection bits

| BTO<2:0> | I <sup>2</sup> Cx Bus Timeout Selection |
|----------|-----------------------------------------|
| 111      | CLC4OUT                                 |
| 110      | CLC3OUT                                 |
| 101      | CLC2OUT                                 |
| 100      | CLC1OUT                                 |
| 011      | TMR6 post scaled output                 |
| 010      | TMR4 post scaled output                 |
| 001      | TMR2 post scaled output                 |
| 000      | Reserved                                |

## REGISTER 33-6: I2CxSTAT0: I<sup>2</sup>C STATUS REGISTER 0

| R-0                 | R-0 | R-0 | R-0                 | R-0 | U-0 | U-0 | U-0   |
|---------------------|-----|-----|---------------------|-----|-----|-----|-------|
| BFRE <sup>(3)</sup> | SMA | MMA | R <sup>(1, 2)</sup> | D   | —   | —   | —     |
| bit 7               |     |     |                     |     |     |     | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

HS = Hardware set

HC = Hardware clear

bit 7 **BFRE:** Bus Free Status bit<sup>(3)</sup>

1 = Indicates the I<sup>2</sup>C bus is Idle

Both SCL and SDA have been high for time-out selected by I2CxCON2<BFRET<1:0>> bits.

I2CxCLK must select a valid clock source for this bit to function.

0 = Bus not Idle (When no I2CxCLK source is selected, this bit remains clear)

bit 6 **SMA:** Slave Module Active Status bit

1 = Set after the 8th falling SCL edge of a received matching 7-bit slave address

Set after the 8th falling SCL edge of a received matching 10-bit slave **low** address

Set after the 8th falling SCL edge of a received matching 10-bit slave **high** w/ read address, only after a previous matching high and low w/ write.

0 = Cleared by any Restart/Stop detected on the bus

Cleared by BTOIF and BCLIF conditions

bit 5 **MMA:** Master Module Active Status bit

1 = Master Mode state machine is active

Set when master state machine asserts a Start on bus

0 = Master state machine is Idle

Cleared when BCLIF is set

Cleared when Stop is shifted out by master.

Cleared for BTOIF condition, after the master successfully shifts out a Stop condition.

bit 4 **R:** Read Information bit <sup>(1, 2)</sup>

1 = Indicates the last matching received (high) address was a Read request

0 = Indicates the last matching received (high) address was a Write

bit 3 **D:** Data bit

1 = Indicates the last byte received or transmitted was data

0 = Indicates the last byte received or transmitted was an address

bit 2-0 **Unimplemented:** Read as 1'b0

**Note 1:** This bit holds the R bit information following the last received address match. Addresses transmitted by the master or appearing on the bus without a match do not affect this bit.

**2:** Clock requests and input from I2CxCLK register are disabled in Slave modes.

**3:** Software must use the EN bit to force master or slave hardware to Idle.

## REGISTER 33-7: I2CxSTAT1: I<sup>2</sup>C STATUS REGISTER 1

| R/W/HS-0            | U-0 | R-1                    | U-0 | R/W/HS-0            | R/S-0/0 | U-0 | R-0                   |
|---------------------|-----|------------------------|-----|---------------------|---------|-----|-----------------------|
| TXWE <sup>(2)</sup> | —   | TXBE <sup>(1, 3)</sup> | —   | RXRE <sup>(2)</sup> | CLRBF   | —   | RXBF <sup>(1,3)</sup> |
| bit 7               |     |                        |     |                     |         |     | bit 0                 |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared | HS = Hardware set HC = Hardware clear                 |

- bit 7 **TXWE:** Transmit Write Error Status bit <sup>(2)</sup>  
1 = A new byte of data was written to I2CxTXB when it was full (Must be cleared by software)  
0 = No transmit write error
- bit 6 **Unimplemented:** Read as '0'
- bit 5 **TXBE:** Transmit Buffer Empty Status bit  
1 = I2CxTXB is empty (Cleared by writing the I2CTXB register)  
0 = I2CxTXB is full
- bit 4 **Unimplemented:** Read as '0'
- bit 3 **RXRE:** Receive Read Error Status bit  
1 = A byte of data was read from I2CxRXB when it was empty. (Must be cleared by software)  
0 = No receive overflow
- bit 2 **CLRBF:** Clear Buffer bit  
Setting this bit clears/empties the receive and transmit buffers, causing reset of RXBF and TXBE.  
Setting this bit clears the I2CxRXIF and I2CxTXIF interrupt flags.  
This bit is set-only special function, and always reads '0'
- bit 1 **Unimplemented:** Read as '0'
- bit 0 **RXBF:** Receive Buffer Full Status bit  
1 = I2CxRXB has received new data (Cleared by reading the I2CxRXB register)  
0 = I2CxRXB is empty

- Note 1:** The bits are held in Reset when EN = 0.
- 2:** Will cause NACK to be sent for slave address and master/slave data read bytes.
- 3:** Used as triggers for DMA operation.

## REGISTER 33-8: I2CxERR: I<sup>2</sup>C ERROR REGISTER

| U-0   | R/W/HS-0               | R/W/HS-0             | R/W/HS-0              | U-0   | R/W-0 | R/W-0 | R/W-0  |
|-------|------------------------|----------------------|-----------------------|-------|-------|-------|--------|
| —     | BTOIF <sup>(1,2)</sup> | BCLIF <sup>(1)</sup> | NACKIF <sup>(1)</sup> | —     | BTOIE | BCLIE | NACKIE |
| bit 7 |                        |                      |                       | bit 0 |       |       |        |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared | HS = Hardware set HC = Hardware clear                 |

- bit 7 **Unimplemented:** Read as '0'
- bit 6 **BTOIF:** Bus Timeout Interrupt Flag bit<sup>(1,2)</sup>  
 1 = Bus Timeout occurred  
 0 = No bus timeout
- bit 5 **BCLIF:** Bus Collision Detect Interrupt Flag bit<sup>(1)</sup>  
 1 = Bus collision detected (On the rising edge of SCL input, SDA output is high and input is sampled low)  
 Slave and Master mode the module immediately goes Idle  
 Multi-Master mode attempts to match slave addresses, and/or goes Idle  
 0 = No bus collision detected
- bit 4 **NACKIF:** NACK Detect Interrupt Flag bit<sup>(1)</sup>  
 1 = When (SMA = 1 || MMA = 1) and a NACK is detected on the bus  
 NACKIF is also set when any of the TXWE, RXRE, TXU, or RXO bits are set.  
 0 = No NACK/Error detected  
 NACKIF is **not** set by the NACK send for non-matching slave addresses
- bit 3 **Unimplemented:** Read as '0'
- bit 2 **BTOIE:** Bus Timeout Interrupt Enable bit  
 1 = Enable interrupt on bus timeout  
 0 = Bus Timeout not enabled
- bit 1 **BCLIE:** Bus Collision Detect Interrupt Enable bit  
 1 = Enable interrupt on bus collision  
 0 = Bus collision interrupts are disabled
- bit 0 **NACKIE:** NACK Detect Interrupt Enable bit  
 1 = Enable interrupt on NACKIF  
 0 = NACKIF interrupt is disabled

**Note 1:** Enabled error interrupt flags are OR'd to produce the PIRx<I2CxEIF> bit.

**2:** User software must select the Bus Timeout Source in the I2CBTO register.

## REGISTER 33-9: I2CxCNT: I<sup>2</sup>C BYTE COUNT REGISTER

|          |         |         |         |         |         |         |         |
|----------|---------|---------|---------|---------|---------|---------|---------|
| R/W-x/u  | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
| CNT<7:0> |         |         |         |         |         |         |         |
| bit 7    |         |         |         | bit 0   |         |         |         |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared | HS = Hardware set    HC = Hardware clear              |

bit 7-0      **CNT<7:0>:** I<sup>2</sup>C Byte Count Register bits

If receiving data,

decremented 8th SCL edge, when a new data byte is loaded into I2CxRXB

If transmitting data,

decremented 9th SCL edge, when a new data byte is moved from I2CxTXB

CNTIF flag is set on 9th falling SCL edge, when I2CxCNT = 0. (Byte count cannot decrement past '0')

**Note 1:** It is recommended to write this register only when the module is IDLE (MMA = 0, SMA = 0) or when clock stretching (CSTR = 1 || MDR = 1).

## REGISTER 33-10: I2CxPIR: I2CxIF INTERRUPT FLAG REGISTER

| R/W/HS-0 | R/W/HS-0 | U-0 | R/W/HS-0 | R/W/HS-0 | R/W/HS-0 | R/W/HS-0 | R/W/HS-0 |
|----------|----------|-----|----------|----------|----------|----------|----------|
| CNTIF    | ACKTIF   | —   | WRIF     | ADRIF    | PCIF     | RSCIF    | SCIF     |
| bit 7    |          |     |          |          |          |          | bit 0    |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

HS = Hardware set

HC = Hardware clear

- bit 7 **CNTIF:** Byte Count Interrupt Flag bit  
1 = When I2CxCNT = 0, set by the 9th falling edge of SCL.  
0 = I2CxCNT condition has not occurred.
- bit 6 **ACKTIF:** Acknowledge Status Time Interrupt Flag bit <sup>(2)</sup> (MODE<2:0> = 0xx OR 11x)  
1 = Set by the 9th falling edge of SCL for any byte when addressed as a slave  
0 = Acknowledge condition not detected.
- bit 5 **Unimplemented:** Read as '0'
- bit 4 **WRIF:** Data Write Interrupt Flag bit (MODE<2:0> = 0xx OR 11x)  
1 = Set the 8th falling edge of SCL for a received data byte.  
0 = Data Write condition not detected
- bit 3 **ADRIF:** Address Interrupt Flag bit (MODE<2:0> = 0xx OR 11x)  
1 = Set the 8th falling edge of SCL for a matching received (high/low) address byte  
0 = Address condition not detected
- bit 2 **PCIF:** Stop Condition Interrupt Flag  
1 = Set on detection of Stop condition  
0 = No Stop condition detected
- bit 1 **RSCIF:** Restart Condition Interrupt Flag  
1 = Set on detection of Restart condition  
0 = No Restart condition detected
- bit 0 **SCIF:** Start Condition Interrupt Flag  
1 = Set on detection of Start condition  
0 = No Start condition detected

**Note 1:** Enabled interrupt flags are OR'd to produce the PIRx<I2CxIF> bit.

**2:** ACKTIF is not set by a matching, 10-bit, high address byte with the R/W bit clear. It is only set after the matching low address byte is shifted in.

## REGISTER 33-11: I2CxPIE: I2CxIE INTERRUPT AND HOLD ENABLE REGISTER

|       |        |     |       |       |       |       |       |
|-------|--------|-----|-------|-------|-------|-------|-------|
| R/W-0 | R/W-0  | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| CNTIE | ACKTIE | —   | WRIE  | ADRIE | PCIE  | RSCIE | SCIE  |
| bit 7 |        |     |       |       |       |       | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

HS = Hardware set

HC = Hardware clear

- bit 7 **CNTIE:** Byte Count Interrupt Enable bit  
 1 = When CNTIF is set  
 0 = Byte count interrupts are disabled
- bit 6 **ACKTIE:** Acknowledge Interrupt and Hold Enable bit  
 1 = When ACKTIF is set  
     If  $\overline{\text{ACK}}$  is generated, CSTR is also set.  
     If NACK is generated, CSTR is unchanged  
 0 = Acknowledge holding and interrupt is disabled
- bit 5 **Unimplemented:** Read as '0'
- bit 4 **WRIE:** Data Write Interrupt and Hold Enable bit  
 1 = When WRIF is set; CSTR is set  
 0 = Data Write holding and interrupt is disabled
- bit 3 **ADRIE:** Address Interrupt and Hold Enable bit  
 1 = When ADRIF is set; CSTR is set  
 0 = Address holding and interrupt is disabled
- bit 2 **PCIE:** Stop Condition Interrupt Enable bit  
 1 = Enable interrupt on detection of Stop condition  
 0 = Stop detection interrupts are disabled
- bit 1 **RSCIE:** Restart Condition Interrupt Enable bit  
 1 = Enable interrupt on detection of Restart condition  
 0 = Start detection interrupts are disabled
- bit 0 **SCIE:** Start Condition Interrupt Enable bit  
 1 = Enable interrupt on detection of Start condition  
 0 = Start detection interrupts are disabled

**Note 1:** Enabled interrupt flags are OR'd to produce the PIRx<I2CxIF> bit.

# PIC18(L)F25/26K83

## REGISTER 33-12: I2CxADR0: I<sup>2</sup>C ADDRESS 0 REGISTER

|       |       |       |       |       |       |       |       |
|-------|-------|-------|-------|-------|-------|-------|-------|
| R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
| ADR7  | ADR6  | ADR5  | ADR4  | ADR3  | ADR2  | ADR1  | ADR0  |
| bit 7 |       |       |       |       |       |       | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

HS = Hardware set

HC = Hardware clear

bit 7-0

**ADR<7-0>**: Address 0 bits

MODE<2:0> = 00x | 11x - 7-bit Slave/Multi-Master Modes

**ADR0<7:1>**: 7-bit Slave Address

**ADR0<0>**: Unused in this mode; bit state is a "don't care"

MODE<2:0> = 01x - 10-bit Slave Modes

**ADR0<7:0>**: Eight Least Significant bits of 10-bit address 0



## REGISTER 33-13: I2CxADR1: I<sup>2</sup>C ADDRESS 1 REGISTER

|       |       |       |       |       |       |       |       |
|-------|-------|-------|-------|-------|-------|-------|-------|
| R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | U-0   |
| ADR14 | ADR13 | ADR12 | ADR11 | ADR10 | ADR9  | ADR8  | —     |
| bit 7 |       |       |       |       |       |       | bit 0 |
| R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | U-0   |
| ADR7  | ADR6  | ADR5  | ADR4  | ADR3  | ADR2  | ADR1  | —     |
| bit 7 |       |       |       |       |       |       | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

HS = Hardware set

HC = Hardware clear

bit 7-1

**ADR[7-1]:** Address 1 bits

MODE<2:0> = 000 | 110 - 7-bit Slave/Multi-Master Modes

**ADR<7:1>:** 7-bit Slave Address

MODE<2:0> = 001 | 111 - 7-bit Slave/Multi-Master modes w/Masking

**ADR<7:1>:** 7-bit Slave Address Mask

MODE<2:0> = 01x - 10-bit Slave Modes

**ADR<14-10>:** Bit pattern sent by master is fixed by I<sup>2</sup>C specification and must be equal to '11110'. However, these bit values are compared by hardware to the received data to determine a match. It is up to the user to set these bits as '11110'.

**ADR<9-8>:** Two Most Significant bits of 10-bit address

bit 0

**Unimplemented:** Read as '0'.

## REGISTER 33-14: I2CxADR2: I<sup>2</sup>C ADDRESS 2 REGISTER

|       |       |       |       |       |       |       |       |
|-------|-------|-------|-------|-------|-------|-------|-------|
| R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
| ADR7  | ADR6  | ADR5  | ADR4  | ADR3  | ADR2  | ADR1  | ADR0  |
| bit 7 |       |       |       |       |       |       | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

HS = Hardware set

HC = Hardware clear

bit 7-0

**ADR<7-0>:** Address 2 bits

MODE<2:0> = 000 | 110 - 7-bit Slave/Multi-Master Modes

**ADR<7:1>:** 7-bit Slave Address

MODE<2:0> = 001 | 111 - 7-bit Slave/Multi-Master Modes with Masking

**ADR<7:1>:** 7-bit Slave Address

MODE<2:0> = 010 - 10-Bit Slave Mode

**ADR<7:0>:** Eight Least Significant bits of second 10-bit address

MODE<2:0> = 011 - 10-Bit Slave Mode with Masking

**ADR<7-0>:** Eight Least Significant bits of 10-bit address mask

## REGISTER 33-15: I2CXADR3: I<sup>2</sup>C ADDRESS 3 REGISTER

|        |       |       |       |       |       |       |       |
|--------|-------|-------|-------|-------|-------|-------|-------|
| R/W-1  | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | U-0   |
| ADR14  | ADR13 | ADR12 | ADR11 | ADR10 | ADR9  | ADR8  | —     |
| bit 15 |       |       |       |       |       |       | bit 8 |
| R/W-1  | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | U-0   |
| ADR7   | ADR6  | ADR5  | ADR4  | ADR3  | ADR2  | ADR1  | —     |
| bit 7  |       |       |       |       |       |       | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

HS = Hardware set

HC = Hardware clear

bit 7-1

**ADR<7-1>:** Address 3 bits

MODE<2:0> = 000 | 110 - 7-bit Slave/Multi-Master Modes

**ADR<7:1>:** 7-bit Slave Address

MODE<2:0> = 001 | 111 - 7-bit Slave/Multi-Master Mode with Masking

**ADR<7:1>:** 7-bit Slave Address

MODE<2:0> = 010 - 10-Bit Slave Mode

**ADR<14-10>:** Bit pattern sent by master is fixed by I<sup>2</sup>C specification and must be equal to '11110'. However, these bit values are compared by hardware to the received data to determine a match. It is up to the user to set these bits as '11110'

**ADR<9-8>:** Two Most Significant bits of 10-bit address

MODE<2:0> = 011 - 10-Bit Slave Mode with Masking

**ADR<14-8>:** 10-bit high address mask

bit 0

**Unimplemented:** Read as '0'

## REGISTER 33-16: I2CxADB0: I<sup>2</sup>C ADDRESS DATA BUFFER 0 REGISTER<sup>(1)</sup>

| R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
|---------|---------|---------|---------|---------|---------|---------|---------|
| ADB7    | ADB6    | ADB5    | ADB4    | ADB3    | ADB2    | ADB1    | ADB0    |
| bit 7   |         |         |         |         |         |         | bit 0   |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared | HS = Hardware set    HC = Hardware clear              |

bit 7-0

MODE<2:0> = 00x

**ADB<7:1>:** Address Data byte

Received matching 7-bit slave address data

**R/W:** Read/not-Write Data bit

Received read/write value from 7-bit address byte

MODE<2:0> = 01x

**ADB<7:0>:** Address Data byte

Received matching lower eight bits of 10-bit slave address data

MODE<2:0> = 100

Unused in this mode; bit state is a "don't care"

MODE<2:0> = 101

**ADB<7:0>:** Low Address Data byte

Low 10-bit address value copied to transmit shift register

MODE<2:0> = 11x

**ADB<7:1>:** Address Data byte

Received matching 7-bit slave address

**R/W:** Read/not-Write Data bit

Received read/write value received 7-bit slave address byte

**Note 1:** This register is read only except in master, 10-bit Address mode (MODE<2:0> = 101).

## REGISTER 33-17: I2CxADB1: I<sup>2</sup>C ADDRESS DATA BUFFER 1 REGISTER<sup>(1)</sup>

| R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
|---------|---------|---------|---------|---------|---------|---------|---------|
| ADB7    | ADB6    | ADB5    | ADB4    | ADB3    | ADB2    | ADB1    | ADB0    |
| bit 7   |         |         |         |         |         |         | bit 0   |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared | HS = Hardware set    HC = Hardware clear              |

bit 7-0

MODE<2:0> = 00x

Unused in this mode; bit state is a "don't care"

MODE<2:0> = 01x

**ADB<7:1>:** 10-bit Address High byte

Received matching 10-bit high address data

**R/W:** Read/not-Write Data bit

Received read/write value from matching 10-bit high address

MODE<2:0> = 100

**ADB<7:1>:** Address Data byte

7-bit address value copied to transmit shift register

**R/W:** Read/not-Write Data bit

Read/write value copied to transmit shift register

MODE<2:0> = 101

**ADB<7:1>:** 10-bit Address High Data byte

10-bit high address value copied to transmit shift register

**R/W:** Read/not-Write Data bit

Read/write value copied to transmit shift register

MODE<2:0> = 11x

**ADB<7:1>:** Address Data byte

7-bit address value copied to transmit shift register

**R/W:** Read/not-Write Data bit

Read/write value copied to transmit shift register

**Note 1:** This register is read only in slave, 7-bit Addressing modes (MODE<2:0> = 0xx)

**TABLE 33-18: SUMMARY OF REGISTERS FOR I<sup>2</sup>C 8-BIT MACRO**

| Name      | Bit 7    | Bit 6  | Bit 5   | Bit 4  | Bit 3      | Bit 2     | Bit 1      | Bit 0  | Register on page |
|-----------|----------|--------|---------|--------|------------|-----------|------------|--------|------------------|
| I2CxBTO   | —        | —      | —       | —      | —          | BTO<2:0>  |            |        | 569              |
| I2CxCLK   | —        | —      | —       | —      | —          | CLK<2:0>  |            |        | 568              |
| I2CxPIE   | CNTIE    | ACKTIE | —       | WRIE   | ADRIE      | PCIE      | RSCIE      | SCIE   | 575              |
| I2CxPIR   | CNTIF    | ACKTIF | —       | WRIF   | ADRIF      | PCIF      | RSCIF      | SCIF   | 574              |
| I2CxERR   | —        | BTOIF  | BCLIF   | NACKIF | —          | BTOIE     | BCLIE      | NACKIE | 572              |
| I2CxSTAT0 | BFRE     | SMA    | MMA     | R      | D          | —         | —          | —      | 570              |
| I2CxSTAT1 | TXWE     | —      | TXBE    | —      | RXRE       | CLRBF     | —          | RXBF   | 571              |
| I2CxCON0  | EN       | RSEN   | S       | CSTR   | MDR        | MODE<2:0> |            |        | 564              |
| I2CxCON1  | ACKCNT   | ACKDT  | ACKSTAT | ACKT   | —          | RXO       | TXU        | CSD    | 566              |
| I2CxCON2  | ACNT     | GCEN   | FME     | ABD    | SDAHT<3:2> |           | BFRET<1:0> |        | 567              |
| I2CxADR0  | ADR<7:0> |        |         |        |            |           |            |        | 576              |
| I2CxADR1  | ADR<7:1> |        |         |        |            |           |            | —      | 577              |
| I2CxADR2  | ADR<7:0> |        |         |        |            |           |            |        | 578              |
| I2CxADR3  | ADR<7:1> |        |         |        |            |           |            | —      | 579              |
| I2CxADB0  | ADB<7:0> |        |         |        |            |           |            |        | 580              |
| I2CxADB1  | ADB<7:0> |        |         |        |            |           |            |        | 581              |
| I2CxCNT   | CNT<7:0> |        |         |        |            |           |            |        | 573              |
| I2CxRXB   | RXB<7:0> |        |         |        |            |           |            |        | —                |
| I2CXTXB   | TXB<7:0> |        |         |        |            |           |            |        | —                |

**Legend:** — = unimplemented, read as '0'. Shaded cells are unused by the I<sup>2</sup>C module.

## 34.0 CAN MODULE

This family of devices contain a Controller Area Network (CAN) module. The CAN module is fully backwards-compatible with the CAN and ECAN modules found in older PIC18 devices.

The Controller Area Network (CAN) module is a serial interface which is useful for communicating with other peripherals or microcontroller devices. This interface, or protocol, was designed to allow communications within noisy environments.

The CAN module is a communication controller, implementing the CAN 2.0A or B protocol as defined in the BOSCH specification. The module will support CAN 1.2, CAN 2.0A, CAN 2.0B Passive and CAN 2.0B Active versions of the protocol. The module implementation is a full CAN system; however, the CAN specification is not covered within this data sheet. Refer to the BOSCH CAN specification for further details.

The module features are as follows:

- Implementation of the CAN protocol, CAN 1.2, CAN 2.0A and CAN 2.0B
- DeviceNet™ data bytes filter support
- Standard and extended data frames
- 0-8 bytes data length
- Programmable bit rate up to 1 Mbit/sec
- Fully backward compatible with CAN modules on older PIC18 devices
- Three modes of operation:
  - Mode 0 – Legacy mode
  - Mode 1 – Enhanced Legacy mode with DeviceNet support
  - Mode 2 – FIFO mode with DeviceNet support
- Support for remote frames with automated handling
- Double-buffered receiver with two prioritized received message storage buffers
- Six buffers programmable as RX and TX message buffers
- 16 full (standard/extended identifier) acceptance filters that can be linked to one of four masks
- Two full acceptance filter masks that can be assigned to any filter
- One full acceptance filter that can be used as either an acceptance filter or acceptance filter mask
- Three dedicated transmit buffers with application specified prioritization and abort capability
- Programmable wake-up functionality with integrated low-pass filter
- Programmable Loopback mode supports self-test operation
- Signaling via interrupt capabilities for all CAN receiver and transmitter error states
- Programmable clock source
- Programmable link to timer module for time-stamping and network synchronization
- Low-power Sleep mode

## 34.1 Module Overview

The CAN bus module consists of a protocol engine and message buffering and control. The CAN protocol engine automatically handles all functions for receiving and transmitting messages on the CAN bus. Messages are transmitted by first loading the appropriate data registers. Status and errors can be checked by reading the appropriate registers. Any message detected on the CAN bus is checked for errors and then matched against filters to see if it should be received and stored in one of the two receive registers.

The CAN module supports the following frame types:

- Standard Data Frame
- Extended Data Frame
- Remote Frame
- Error Frame
- Overload Frame Reception

The CANRX input pin is selected with the CANRXPPS register. The CANTX output pin is selected with each pin's RxyPPS register.

**Note:** The CANRX pin defaults to pin RB3, but the CANTX has no default location and must be assigned to a pin before CAN transmissions can occur.

In Normal mode, the user must ensure that the appropriate TRIS bit for CANRX is set and the appropriate TRIS bit for CANRX is cleared. In addition, the appropriate ANSEL bit for CANRX must be cleared to disable the analog input buffer.

**Note:** Unlike older Microchip devices with CAN functionality, the CAN pins can be mapped to pins with analog functionality. Ensure that the analog functionality on the CANRX pin is disabled, or the CAN module will not properly function.

## 34.1.1 MODULE FUNCTIONALITY

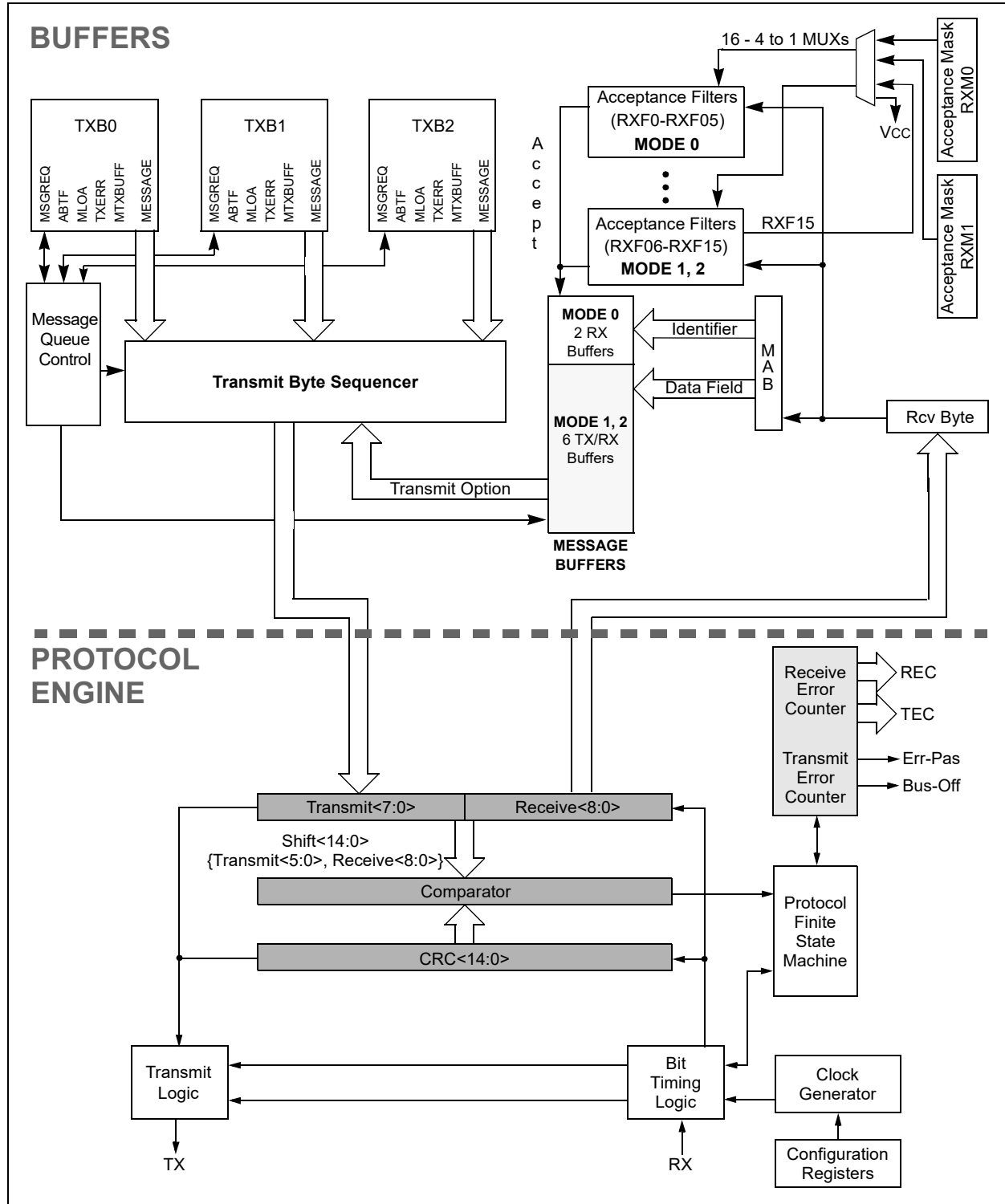
The CAN bus module consists of a protocol engine, message buffering and control (see [Figure 34-1](#)). The protocol engine can best be understood by defining the types of data frames to be transmitted and received by the module.

The following sequence illustrates the necessary initialization steps before the CAN module can be used to transmit or receive a message. Steps can be added or removed depending on the requirements of the application.

1. Use the CANRXPPS and appropriate RxyPPS registers to map the CANRX and CANTX functions to the desired pins of the device.
2. Initialize LAT, TRIS and ANSEL bits for the selected CANRX and CANTX pins.
3. Ensure that the CAN module is in Configuration mode.
4. Select CAN Functional mode.
5. Set up the Baud Rate registers.
6. Set up the Filter and Mask registers.
7. Set the CAN module to Normal mode or any other mode required by the application logic.



**FIGURE 34-1: CAN BUFFERS AND PROTOCOL ENGINE BLOCK DIAGRAM**



## 34.2 CAN Modes of Operation

The CAN module has six main modes of operation:

- Configuration mode
- Disable/Sleep mode
- Normal Operation mode
- Listen Only mode
- Loopback mode
- Error Recognition mode

All modes, except Error Recognition, are requested by setting the REQOP bits (CANCON<7:5>). Error Recognition mode is requested through the RXM bits of the Receive Buffer register(s). Entry into a mode is acknowledged by monitoring the OPMODE bits.

When changing modes, the mode will not actually change until all pending message transmissions are complete. Because of this, the user must verify that the device has actually changed into the requested mode before further operations are executed.

**Note:** The module may fail to change modes from Configuration mode if the CANRX and CANTX pins are not externally connected to a CAN transceiver. If connection to a transceiver is not desired for the particular use case or application (for example, transitioning to Loopback mode for development/debugging), the CANRX pin must be externally tied to VDD through a 10k pull-up resistor.

### 34.2.1 CONFIGURATION MODE

The CAN module has to be initialized before the activation. This is only possible if the module is in the Configuration mode. The Configuration mode is requested by setting the REQOP<2:0> bits to 0b100. Only when the Status bits OPMODE<2:0> are equal to 0b100, can the initialization be performed. Afterwards, the Configuration registers, the acceptance mask registers and the acceptance filter registers can be written.

Configuration mode protects the user from accidentally violating the CAN protocol through programming errors, as all registers which control the configuration of the module can not be modified while the module is on-line. The CAN module will not enter the Configuration mode while a transmission or reception is taking place. The following registers can only be modified in Configuration mode:

- Configuration Registers
- Functional Mode Selection Registers
- Bit Timing Registers
- Identifier Acceptance Filter Registers
- Identifier Acceptance Mask Registers
- Filter and Mask Control Registers
- Mask Selection Registers

In the Configuration mode, the module will not transmit or receive. The error counters are cleared and the interrupt flags remain unchanged. The programmer will have access to Configuration registers that are access restricted in other modes. I/O pins will revert to normal I/O functions.

### 34.2.2 DISABLE/SLEEP MODE

When the REQOP<2:0> bits are set to '001', the module will enter Disable/Sleep mode. This mode is similar to disabling other peripheral modules by turning off the module enables. This causes the module internal clock to stop unless the module is active (i.e., receiving or transmitting a message). If the module is active, the module will wait for 11 recessive bits on the CAN bus, detect that condition as an Idle bus, then accept the module Disable/Sleep command. OPMODE<2:0> = 001 indicates whether the module successfully went into the module Disable/Sleep mode. In Disable/Sleep mode, the module will not transmit or receive. The module has the ability to set the WAKIF bit due to bus activity. However, any pending interrupts will remain and the error counters will retain their value.

The WAKIF interrupt is the only module interrupt that is still active in the Disable/Sleep mode. If the WAKDIS is cleared and WAKIE is set, the processor will receive an interrupt whenever the module detects a recessive to dominant transition. On wake-up, the module will automatically be set to the previous mode of operation. For example, if the module was switched from Normal to Disable/Sleep mode on bus activity wake-up, the module will automatically enter into Normal mode and the first message that caused the module to wake-up is lost. The module will not generate any error frame. Firmware logic must detect this condition and make sure that retransmission is requested. If the processor receives a wake-up interrupt while it is sleeping, more than one message may get lost. The actual number of messages lost would depend on the processor oscillator start-up time and incoming message bit rate.

The CANTX pin will stay in the recessive state while the module is in Disable/Sleep mode.

### 34.2.3 NORMAL MODE

This is the standard operating mode of the CAN module. In this mode, the device actively monitors all bus messages and generates Acknowledge bits, error frames, etc. This is also the only mode in which the CAN module will transmit messages over the CAN bus. The Normal mode is activated by clearing the mode request bits in the CANCON register.

## 34.2.4 LISTEN ONLY MODE

Listen Only mode provides a means for the CAN module to receive all messages, including messages with errors. This mode can be used for bus monitor applications or for detecting the baud rate in 'hot plugging' situations. For auto-baud detection, it is necessary that there are at least two other nodes which are communicating with each other. The baud rate can be detected empirically by testing different values until valid messages are received. The Listen Only mode is a silent mode, meaning no messages will be transmitted while in this state, including error flags or Acknowledge signals. In Listen Only mode, both valid and invalid messages will be received, regardless of RXMn bit settings. The filters and masks can still be used to allow only particular valid messages to be loaded into the Receive registers, or the filter masks can be set to all zeros to allow a message with any identifier to pass. All invalid messages will be received in this mode, regardless of filters and masks or RXMn Receive Buffer mode bits. The error counters are reset and deactivated in this state. The Listen Only mode is activated by setting the mode request bits in the CANCON register to 0b011.

## 34.2.5 LOOPBACK MODE

This mode will allow internal transmission of messages from the transmit buffers to the receive buffers without actually transmitting messages on the CAN bus. This mode can be used in system development and testing. In this mode, the ACK bit is ignored and the device will allow incoming messages from itself, just as if they were coming from another node. The Loopback mode is a silent mode, meaning no messages will be transmitted while in this state, including error flags or Acknowledge signals. The TXCAN pin will revert to port I/O while the device is in this mode. The filters and masks can be used to allow only particular messages to be loaded into the receive registers. The masks can be set to all zeros to provide a mode that accepts all messages. The Loopback mode is activated by setting the mode request bits in the CANCON register to 0b010.

## 34.2.6 ERROR RECOGNITION MODE

The module can be set to ignore all errors and receive any message. In functional Mode 0, the Error Recognition mode is activated by setting the RXM<1:0> bits in the RXBnCON registers to '11'. In this mode, the data which is in the message assembly buffer until the error time, is copied in the receive buffer and can be read via the CPU interface.

## 34.3 CAN Module Functional Modes

In addition to CAN modes of operation, the CAN module offers a total of three functional modes. Each of these modes are identified as Mode 0, Mode 1 and Mode 2.

### 34.3.1 MODE 0 – LEGACY MODE

Mode 0 is designed to be fully compatible with CAN modules used in PIC18CXX8 and PIC18FXX8 devices. This is the default mode of operation on all Reset conditions. As a result, module code written for the PIC18XX8 CAN module may be used on the CAN module with only very minor code changes.

The following is the list of resources available in Mode 0:

- Three transmit buffers: TXB0, TXB1 and TXB2
- Two receive buffers: RXB0 and RXB1
- Two acceptance masks, one for each receive buffer: RXM0, RXM1
- Six acceptance filters, 2 for RXB0 and 4 for RXB1: RXF0, RXF1, RXF2, RXF3, RXF4, RXF5

### 34.3.2 MODE 1 – ENHANCED LEGACY MODE

Mode 1 is similar to Mode 0, with the exception that more resources are available in Mode 1. There are 16 acceptance filters and two acceptance mask registers. Acceptance Filter 15 can be used as either an acceptance filter or an acceptance mask register. In addition to three transmit and two receive buffers, there are six more message buffers. One or more of these additional buffers can be programmed as transmit or receive buffers. These additional buffers can also be programmed to automatically handle RTR messages.

Fourteen of sixteen acceptance filter registers can be dynamically associated to any receive buffer and acceptance mask register. One can use this capability to associate more than one filter to any one buffer.

When a receive buffer is programmed to use standard identifier messages, part of the full acceptance filter register can be used as a data byte filter. The length of the data byte filter is programmable from 0 to 18 bits. This functionality simplifies implementation of high-level protocols, such as the DeviceNet™ protocol.

The following is the list of resources available in Mode 1:

- Three transmit buffers: TXB0, TXB1 and TXB2
- Two receive buffers: RXB0 and RXB1
- Six buffers programmable as TX or RX: B0-B5
- Automatic RTR handling on B0-B5
- Sixteen dynamically assigned acceptance filters: RXF0-RXF15
- Two dedicated acceptance mask registers; RXF15 programmable as third mask: RXM0-RXM1, RXF15
- Programmable data filter on standard identifier messages: SDFLC

## 34.3.3 MODE 2 – ENHANCED FIFO MODE

In Mode 2, two or more receive buffers are used to form the receive FIFO (first in, first out) buffer. There is no one-to-one relationship between the receive buffer and acceptance filter registers. Any filter that is enabled and linked to any FIFO receive buffer can generate acceptance and cause FIFO to be updated.

FIFO length is user-programmable, from 2-8 buffers deep. FIFO length is determined by the very first programmable buffer that is configured as a transmit buffer. For example, if Buffer 2 (B2) is programmed as a transmit buffer, FIFO consists of RXB0, RXB1, B0 and B1, creating a FIFO length of four. If all programmable buffers are configured as receive buffers, FIFO will have the maximum length of eight.

The following is the list of resources available in Mode 2:

- Three transmit buffers: TXB0, TXB1 and TXB2
- Two receive buffers: RXB0 and RXB1
- Six buffers programmable as TX or RX; receive buffers form FIFO: B0-B5
- Automatic RTR handling on B0-B5
- Sixteen acceptance filters: RXF0-RXF15
- Two dedicated acceptance mask registers; RXF15 programmable as third mask: RXM0-RXM1, RXF15
- Programmable data filter on standard identifier messages: SDFLC, useful for DeviceNet protocol

## 34.4 CAN Message Buffers

### 34.4.1 DEDICATED TRANSMIT BUFFERS

The CAN module implements three dedicated transmit buffers – TXB0, TXB1 and TXB2. Each of these buffers occupies 14 bytes of SRAM and are mapped into the SFR memory map. These are the only transmit buffers available in Mode 0. Mode 1 and 2 may access these and other additional buffers.

Each transmit buffer contains one Control register (TXBnCON), four Identifier registers (TXBnSIDL, TXBnSIDH, TXBnEIDL, TXBnEIDH), one Data Length Count register (TXBnDLC) and eight Data Byte registers (TXBnDm).

### 34.4.2 DEDICATED RECEIVE BUFFERS

The CAN module implements two dedicated receive buffers: RXB0 and RXB1. Each of these buffers occupies 14 bytes of SRAM and are mapped into SFR memory map. These are the only receive buffers available in Mode 0. Mode 1 and 2 may access these and other additional buffers.

Each receive buffer contains one Control register (RXBnCON), four Identifier registers (RXBnSIDL, RXBnSIDH, RXBnEIDL, RXBnEIDH), one Data Length Count register (RXBnDLC) and eight Data Byte registers (RXBnDm).

There is also a separate Message Assembly Buffer (MAB) which acts as an additional receive buffer. MAB is always committed to receiving the next message from the bus and is not directly accessible to user firmware. The MAB assembles all incoming messages one by one. A message is transferred to appropriate receive buffers only if the corresponding acceptance filter criteria is met.

### 34.4.3 PROGRAMMABLE TRANSMIT/RECEIVE BUFFERS

The CAN module implements six non-dedicated buffers: B0-B5. These buffers are individually programmable as either transmit or receive buffers. These buffers are available only in Mode 1 and 2. As with dedicated transmit and receive buffers, each of these programmable buffers occupies 14 bytes of SRAM and are mapped into SFR memory map.

Each buffer contains one Control register (BnCON), four Identifier registers (BnSIDL, BnSIDH, BnEIDL, BnEIDH), one Data Length Count register (BnDLC) and eight Data Byte registers (BnDm). Each of these registers contains two sets of control bits. Depending on whether the buffer is configured as transmit or receive, one would use the corresponding control bit set. By default, all buffers are configured as receive buffers. Each buffer can be individually configured as a transmit or receive buffer by setting the corresponding TXENn bit in the BSEL0 register.

When configured as transmit buffers, user firmware may access transmit buffers in any order similar to accessing dedicated transmit buffers. In receive configuration with Mode 1 enabled, user firmware may also access receive buffers in any order required. But in Mode 2, all receive buffers are combined to form a single FIFO. Actual FIFO length is programmable by user firmware. Access to FIFO must be done through the FIFO Pointer bits (FP<4:0>) in the CANCON register. It must be noted that there is no hardware protection against out of order FIFO reads.

### 34.4.4 PROGRAMMABLE AUTO-RTR BUFFERS

In Mode 1 and 2, any of six programmable transmit/receive buffers may be programmed to automatically respond to predefined RTR messages without user firmware intervention. Automatic RTR handling is enabled by setting the TX2EN bit in the BSEL0 register and the RTREN bit in the BnCON register. After this setup, when an RTR request is received, the TXREQ bit is automatically set and the current buffer content is automatically queued for transmission as a RTR response. As with all transmit buffers, once the TXREQ bit is set, buffer registers become read-only and any writes to them will be ignored.

The following outlines the steps required to automatically handle RTR messages:

1. Set buffer to Transmit mode by setting the TXnEN bit to '1' in the BSEL0 register.
2. At least one acceptance filter must be associated with this buffer and preloaded with the expected RTR identifier.
3. Bit, RTREN in the BnCON register, must be set to '1'.
4. Buffer must be preloaded with the data to be sent as a RTR response.

Normally, user firmware will keep buffer data registers up to date. If firmware attempts to update the buffer while an automatic RTR response is in the process of transmission, all writes to buffers are ignored.

## 34.5 CAN Message Transmission

### 34.5.1 INITIATING TRANSMISSION

For the MCU to have write access to the message buffer, the TXREQ bit must be clear, indicating that the message buffer is clear of any pending message to be transmitted. At a minimum, the SIDH, SIDL and DLC registers must be loaded. If data bytes are present in the message, the Data registers must also be loaded. If the message is to use extended identifiers, the EIDH:EIDL registers must also be loaded and the EXIDE bit set.

To initiate message transmission, the TXREQ bit must be set for each buffer to be transmitted. When TXREQ is set, the TXABT, TXLARB and TXERR bits will be cleared. To successfully complete the transmission, there must be at least one node with matching baud rate on the network.

Setting the TXREQ bit does not initiate a message transmission; it merely flags a message buffer as ready for transmission. Transmission will start when the device detects that the bus is available. The device will then begin transmission of the highest priority message that is ready.

When the transmission has completed successfully, the TXREQ bit will be cleared, the TXBnIF bit will be set and an interrupt will be generated if the TXBnIE bit is set.

If the message transmission fails, the TXREQ will remain set, indicating that the message is still pending for transmission and one of the following condition flags will be set. If the message started to transmit but encountered an error condition, the TXERR and the IRXIF bits will be set and an interrupt will be generated. If the message lost arbitration, the TXLARB bit will be set.

### 34.5.2 ABORTING TRANSMISSION

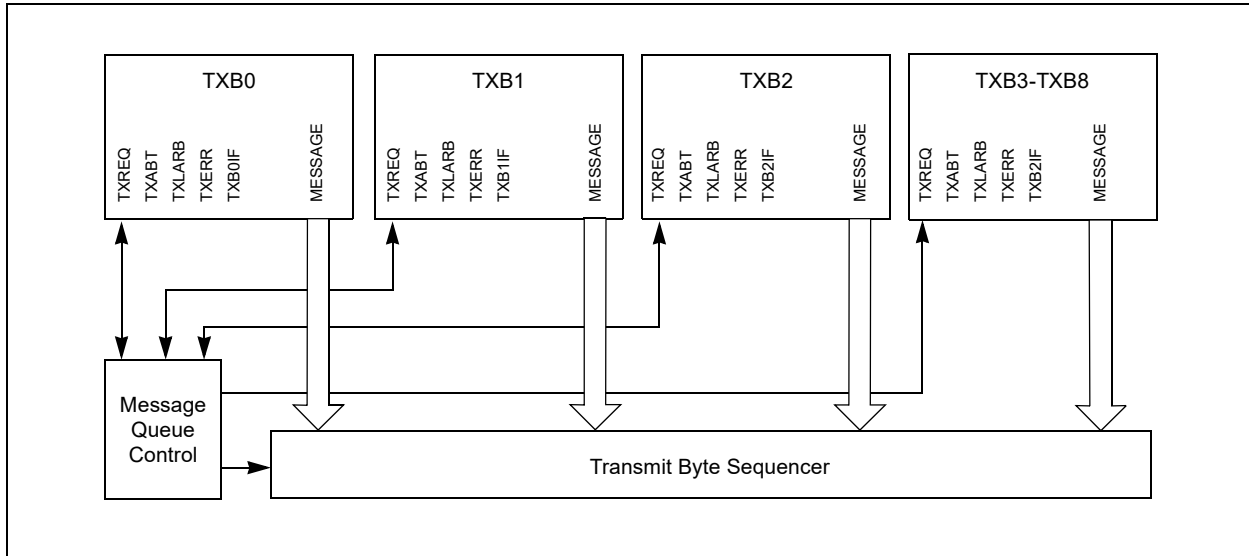
The MCU can request to abort a message by clearing the TXREQ bit associated with the corresponding message buffer (TXBnCON<3> or BnCON<3>). Setting the ABAT bit (CANCON<4>) will request an abort of all pending messages. If the message has not yet started transmission, or if the message started but is interrupted by loss of arbitration or an error, the abort will be processed. The abort is indicated when the module sets the TXABT bit for the corresponding buffer (TXBnCON<6> or BnCON<6>). If the message has started to transmit, it will attempt to transmit the current message fully. If the current message is transmitted fully and is not lost to arbitration or an error, the TXABT bit will not be set because the message was transmitted successfully. Likewise, if a message is being transmitted during an abort request and the message is lost to arbitration or an error, the message will not be retransmitted and the TXABT bit will be set, indicating that the message was successfully aborted.

Once an abort is requested by setting the ABAT or TXABT bits, it cannot be cleared to cancel the abort request. Only CAN module hardware or a POR condition can clear it.

### 34.5.3 TRANSMIT PRIORITY

Transmit priority is a prioritization within the ECAN module of the pending transmittable messages. This is independent from, and not related to, any prioritization implicit in the message arbitration scheme built into the CAN protocol. Prior to sending the Start-of-Frame (SOF), the priority of all buffers that are queued for transmission is compared. The transmit buffer with the highest priority will be sent first. If two buffers have the same priority setting, the buffer with the highest buffer number will be sent first. There are four levels of transmit priority. If the TXP bits for a particular message buffer are set to '11', that buffer has the highest possible priority. If the TXP bits for a particular message buffer are set to '00', that buffer has the lowest possible priority.

**FIGURE 34-2: TRANSMIT BUFFERS**



## 34.6 Message Reception

### 34.6.1 RECEIVING A MESSAGE

Of all receive buffers, the MAB is always committed to receiving the next message from the bus. The MCU can access one buffer while the other buffer is available for message reception or holding a previously received message.

**Note:** The entire contents of the MAB are moved into the receive buffer once a message is accepted. This means that regardless of the type of identifier (standard or extended) and the number of data bytes received, the entire receive buffer is overwritten with the MAB contents. Therefore, the contents of all registers in the buffer must be assumed to have been modified when any message is received.

When a message is moved into either of the receive buffers, the associated RXFUL bit is set. This bit must be cleared by the MCU when it has completed processing the message in the buffer in order to allow a new message to be received into the buffer. This bit provides a positive lockout to ensure that the firmware has finished with the message before the module attempts to load a new message into the receive buffer. If the receive interrupt is enabled, an interrupt will be generated to indicate that a valid message has been received.

Once a message is loaded into any matching buffer, user firmware may determine exactly what filter caused this reception by checking the filter hit bits in the RXBnCON or BnCON registers. In Mode 0, FILHIT<2:0> of RXBnCON serve as filter hit bits. In Mode 1 and 2, FILHIT<4:0> bits of BnCON serve as filter hit bits. The same registers also indicate whether

the current message is an RTR frame or not. A received message is considered a standard identifier message if the EXID/EXIDE bit in the RXBnSIDL or the BnSIDL register is cleared. Conversely, a set EXID bit indicates an extended identifier message. If the received message is a standard identifier message, user firmware needs to read the SIDL and SIDH registers. In the case of an extended identifier message, firmware should read the SIDL, SIDH, EIDL and EIDH registers. If the RXBnDLC or BnDLC register contain non-zero data count, user firmware should also read the corresponding number of data bytes by accessing the RXBnDm or the BnDm registers. When a received message is an RTR, and if the current buffer is not configured for automatic RTR handling, user firmware must take appropriate action and respond manually.

Each receive buffer contains RXM bits to set special Receive modes. In Mode 0, RXM<1:0> bits in RXBnCON define a total of four Receive modes. In Mode 1 and 2, RXM1 bit, in combination with the EXID mask and filter bit, define the same four receive modes. Normally, these bits are set to '00' to enable reception of all valid messages as determined by the appropriate acceptance filters. In this case, the determination of whether or not to receive standard or extended messages is determined by the EXIDE bit in the acceptance filter register. In Mode 0, if the RXM bits are set to '01' or '10', the receiver will accept only messages with standard or extended identifiers, respectively. If an acceptance filter has the EXIDE bit set, such that it does not correspond with the RXM mode, that acceptance filter is rendered useless. In Mode 1 and 2, setting EXID in the SIDL Mask register will ensure that only standard or extended identifiers are received. These two modes of RXM bits can be used in systems where it is known that only standard or extended messages will be on the bus. If the RXM bits are set to '11' (RXM1 = 1 in Mode 1 and 2), the buffer will receive all

messages regardless of the values of the acceptance filters. Also, if a message has an error before the end of frame, that portion of the message assembled in the MAB before the error frame will be loaded into the buffer. This mode may serve as a valuable debugging tool for a given CAN network. It should not be used in an actual system environment as the actual system will always have some bus errors and all nodes on the bus are expected to ignore them.

In Mode 1 and 2, when a programmable buffer is configured as a transmit buffer and one or more acceptance filters are associated with it, all incoming messages matching this acceptance filter criteria will be discarded. To avoid this scenario, user firmware must make sure that there are no acceptance filters associated with a buffer configured as a transmit buffer.

## 34.6.2 RECEIVE PRIORITY

When in Mode 0, RXB0 is the higher priority buffer and has two message acceptance filters associated with it. RXB1 is the lower priority buffer and has four acceptance filters associated with it. The lower number of acceptance filters makes the match on RXB0 more restrictive and implies a higher priority for that buffer. Additionally, the RXB0CON register can be configured such that if RXB0 contains a valid message and another valid message is received, an overflow error will not occur and the new message will be moved into RXB1 regardless of the acceptance criteria of RXB1. There are also two programmable acceptance filter masks available, one for each receive buffer (see [Section 34.4 “CAN Message Buffers”](#)).

In Mode 1 and 2, there are a total of 16 acceptance filters available and each can be dynamically assigned to any of the receive buffers. A buffer with a lower number has higher priority. Given this, if an incoming message matches with two or more receive buffer acceptance criteria, the buffer with the lower number will be loaded with that message.

## 34.6.3 ENHANCED FIFO MODE

When configured for Mode 2, two of the dedicated receive buffers in combination with one or more programmable transmit/receive buffers, are used to create a maximum of an eight buffers deep FIFO buffer. In this mode, there is no direct correlation between filters and receive buffer registers. Any filter that has been enabled can generate an acceptance. When a message has been accepted, it is stored in the next available receive buffer register and an internal Write Pointer is incremented. The FIFO can be a maximum of eight buffers deep. The entire FIFO must consist of contiguous receive buffers. The FIFO head begins at RXB0 buffer and its tail spans toward B5. The maximum length of the FIFO is limited by the presence or absence of the first transmit buffer starting from B0. If a buffer is configured as a transmit buffer, the FIFO length is reduced accordingly. For instance, if B3 is

configured as a transmit buffer, the actual FIFO will consist of RXB0, RXB1, B0, B1 and B2, a total of five buffers. If B0 is configured as a transmit buffer, the FIFO length will be two. If none of the programmable buffers are configured as a transmit buffer, the FIFO will be eight buffers deep. A system that requires more transmit buffers should try to locate transmit buffers at the very end of B0-B5 buffers to maximize available FIFO length.

When a message is received in FIFO mode, the Interrupt Flag Code bits (EICODE<4:0>) in the CANSTAT register will have a value of '10000', indicating the FIFO has received a message. FIFO Pointer bits, FP<3:0> in the CANCON register, point to the buffer that contains data not yet read. The FIFO Pointer bits, in this sense, serve as the FIFO Read Pointer. The user should use the FP bits and read corresponding buffer data. When receive data is no longer needed, the RXFUL bit in the current buffer must be cleared, causing FP<3:0> to be updated by the module.

To determine whether FIFO is empty or not, the user may use the FP<3:0> bits to access the RXFUL bit in the current buffer. If RXFUL is cleared, the FIFO is considered to be empty. If it is set, the FIFO may contain one or more messages. In Mode 2, the module also provides a bit called FIFO High Water Mark (FIFOWM) in the ECANCON register. This bit can be used to cause an interrupt whenever the FIFO contains only one or four empty buffers. The FIFO high water mark interrupt can serve as an early warning to a full FIFO condition.

## 34.6.4 TIME-STAMPING

The CAN module can be programmed to generate a time-stamp for every message that is received. When enabled, the module generates a capture signal for the CCP modules, which in turn captures the value of Timer1, Timer3 or Timer5. This value can be used as the message time-stamp.

To use the time-stamp capability, set the CTS<3:0> bits of the appropriate CCPxCAP register to '1000' to configure the CCP module capture input to the CAN\_rx\_timestamp signal.

In addition, the CAN\_rx\_timestamp can be chosen as a signal input for the Signal Measurement Timer, which can be used for a variety of other timing applications.

## 34.7 Message Acceptance Filters and Masks

The message acceptance filters and masks are used to determine if a message in the Message Assembly Buffer should be loaded into any of the receive buffers. Once a valid message has been received into the MAB, the identifier fields of the message are compared to the filter values. If there is a match, that message will be loaded into the appropriate receive buffer. The filter masks are used to determine which bits in the identifier are examined with the filters. A truth table is shown below in Table 34-1 that indicates how each bit in the identifier is compared to the masks and filters to determine if a message should be loaded into a receive buffer. The mask essentially determines which bits to apply the acceptance filters to. If any mask bit is set to a zero, then that bit will automatically be accepted regardless of the filter bit.

**TABLE 34-1: FILTER/MASK TRUTH TABLE**

| Mask bit n | Filter bit n | Message Identifier bit n001 | Accept or Reject bit n |
|------------|--------------|-----------------------------|------------------------|
| 0          | x            | x                           | Accept                 |
| 1          | 0            | 0                           | Accept                 |
| 1          | 0            | 1                           | Reject                 |
| 1          | 1            | 0                           | Reject                 |
| 1          | 1            | 1                           | Accept                 |

**Legend:** x = don't care

In Mode 0, acceptance filters, RXF0 and RXF1, and filter mask, RXM0, are associated with RXB0. Filters, RXF2, RXF3, RXF4 and RXF5, and mask, RXM1, are associated with RXB1.

In Mode 1 and 2, there are an additional ten acceptance filters, RXF6-RXF15, creating a total of 16 available filters. RXF15 can be used either as an acceptance filter or acceptance mask register. Each of these acceptance filters can be individually enabled or disabled by setting or clearing the RXFENn bit in the RXFCONn register. Any of these 16 acceptance filters can be dynamically associated with any of the receive buffers. Actual association is made by setting the appropriate bits in the RXFBCONn register. Each RXFBCONn register contains a nibble for each filter. This nibble can be used to associate a specific filter to any of available receive buffers. User firmware may associate more than one filter to any one specific receive buffer.

In addition to dynamic filter to buffer association, in Mode 1 and 2, each filter can also be dynamically associated to available Acceptance Mask registers. The FILn\_m bits in the MSELn register can be used to link a specific acceptance filter to an acceptance mask register. As with filter to buffer association, one can also associate more than one mask to a specific acceptance filter.

When a filter matches and a message is loaded into the receive buffer, the filter number that enabled the message reception is loaded into the FILHIT bit(s). In Mode 0 for RXB1, the RXB1CON register contains the FILHIT<2:0> bits. They are coded as follows:

- 101 = Acceptance Filter 5 (RXF5)
- 100 = Acceptance Filter 4 (RXF4)
- 011 = Acceptance Filter 3 (RXF3)
- 010 = Acceptance Filter 2 (RXF2)
- 001 = Acceptance Filter 1 (RXF1)
- 000 = Acceptance Filter 0 (RXF0)

**Note:** '000' and '001' can only occur if the RXB0DBEN bit is set in the RXB0CON register, allowing RXB0 messages to rollover into RXB1.

The coding of the RXB0DBEN bit enables these three bits to be used similarly to the FILHIT bits and to distinguish a hit on filter, RXF0 and RXF1, in either RXB0 or after a rollover into RXB1.

- 111 = Acceptance Filter 1 (RXF1)
- 110 = Acceptance Filter 0 (RXF0)
- 001 = Acceptance Filter 1 (RXF1)
- 000 = Acceptance Filter 0 (RXF0)

If the RXB0DBEN bit is clear, there are six codes corresponding to the six filters. If the RXB0DBEN bit is set, there are six codes corresponding to the six filters, plus two additional codes corresponding to RXF0 and RXF1 filters, that rollover into RXB1.

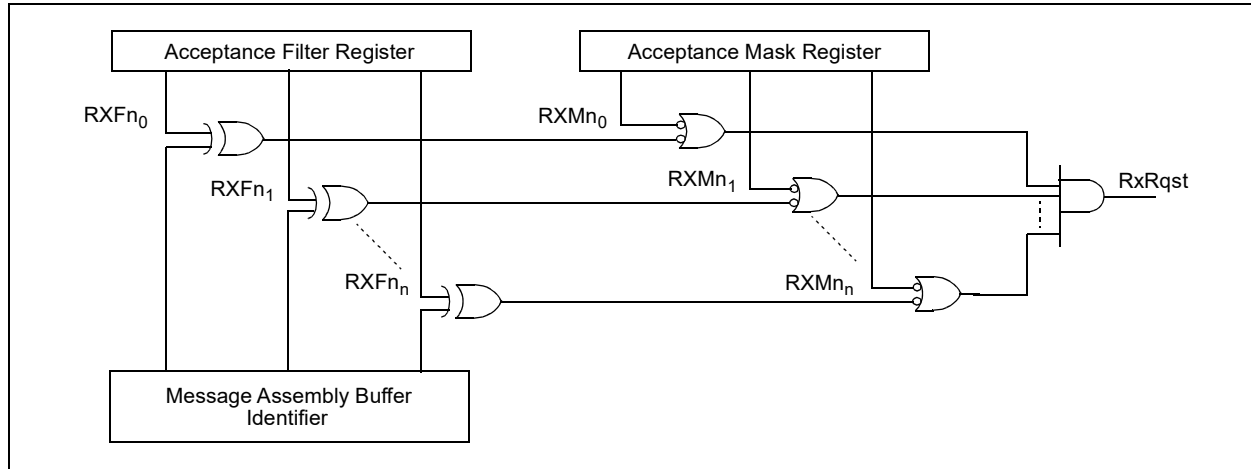
In Mode 1 and 2, each buffer control register contains five bits of filter hit bits (FILHIT<4:0>). A binary value of '0' indicates a hit from RXF0 and 15 indicates RXF15.

If more than one acceptance filter matches, the FILHIT bits will encode the binary value of the lowest numbered filter that matched. In other words, if filter RXF2 and filter RXF4 match, FILHIT will be loaded with the value for RXF2. This essentially prioritizes the acceptance filters with a lower number filter having higher priority. Messages are compared to filters in ascending order of filter number.

The mask and filter registers can only be modified when the CAN module is in Configuration mode.



**FIGURE 34-3: MESSAGE ACCEPTANCE MASK AND FILTER OPERATION**



## 34.8 Baud Rate Setting

All nodes on a given CAN bus must have the same nominal bit rate. The CAN protocol uses Non-Return-to-Zero (NRZ) coding which does not encode a clock within the data stream. Therefore, the receive clock must be recovered by the receiving nodes and synchronized to the transmitter's clock.

As oscillators and transmission time may vary from node to node, the receiver must have some type of Phase Lock Loop (PLL) synchronized to data transmission edges to synchronize and maintain the receiver clock. Since the data is NRZ coded, it is necessary to include bit stuffing to ensure that an edge occurs at least every six bit times to maintain the Digital Phase Lock Loop (DPLL) synchronization.

The bit timing of the CAN module is implemented using a DPLL that is configured to synchronize to the incoming data and provides the nominal timing for the transmitted data. The DPLL breaks each bit time into multiple segments made up of minimal periods of time called the *Time Quanta* (TQ).

Bus timing functions executed within the bit time frame, such as synchronization to the local oscillator, network transmission delay compensation and sample point positioning, are defined by the programmable bit timing logic of the DPLL.

All devices on the CAN bus must use the same bit rate. However, all devices are not required to have the same master oscillator clock frequency. For the different clock frequencies of the individual devices, the bit rate has to be adjusted by appropriately setting the baud rate prescaler and number of time quanta in each segment.

The "Nominal Bit Rate" is the number of bits transmitted per second, assuming an ideal transmitter with an ideal oscillator, in the absence of resynchronization. The nominal bit rate is defined to be a maximum of 1 Mb/s.

The "Nominal Bit Time" is defined as:

### EQUATION 34-1: NOMINAL BIT TIME

$$T_{BIT} = 1/\text{Nominal Bit Rate}$$

The Nominal Bit Time can be thought of as being divided into separate, non-overlapping time segments. These segments (Figure 34-4) include:

- Synchronization Segment (Sync\_Seg)
- Propagation Time Segment (Prop\_Seg)
- Phase Buffer Segment 1 (Phase\_Seg1)
- Phase Buffer Segment 2 (Phase\_Seg2)

The time segments (and thus, the Nominal Bit Time) are, in turn, made up of integer units of time called Time Quanta or TQ (see Figure 34-4). By definition, the Nominal Bit Time is programmable from a minimum of 8 TQ to a maximum of 25 TQ. Also by definition, the minimum Nominal Bit Time is 1  $\mu$ s, corresponding to a maximum 1 Mb/s rate. The actual duration is given by the following relationship:

### EQUATION 34-2: NOMINAL BIT TIME DURATION

$$\text{Nominal Bit Time} = TQ * (\text{Sync\_Seg} + \text{Prop\_Seg} + \text{Phase\_Seg1} + \text{Phase\_Seg2})$$

The Time Quantum is a fixed unit derived from the oscillator period. It is also defined by the programmable baud rate prescaler, with integer values from 1 to 64, in addition to a fixed divide-by-two for clock generation. Mathematically, this is:

## EQUATION 34-3: TIME QUANTUM

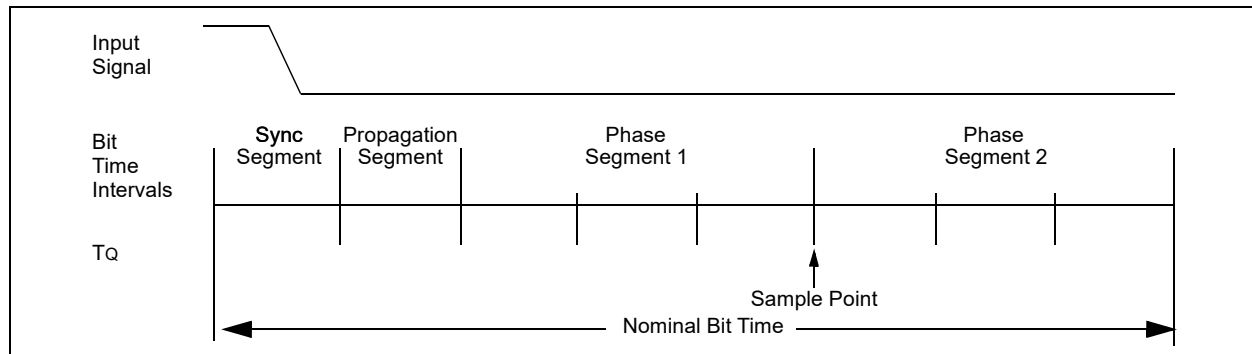
$$TQ (\mu s) = (2 * (BRP + 1)) / FOSC (MHz)$$

or

$$TQ (\mu s) = (2 * (BRP + 1)) * TOSC (\mu s)$$

where FOSC is the clock frequency, TOSC is the corresponding oscillator period and BRP is an integer (0 through 63) represented by the binary values of BRGCON1<5:0>. The equation above refers to the effective clock frequency used by the microcontroller. If, for example, a 10 MHz crystal in HS mode is used, then FOSC = 10 MHz and TOSC = 100 ns. If the same 10 MHz crystal is used in HS-PLL mode, then the effective frequency is FOSC = 40 MHz and TOSC = 25 ns.

**FIGURE 34-4: BIT TIME PARTITIONING**



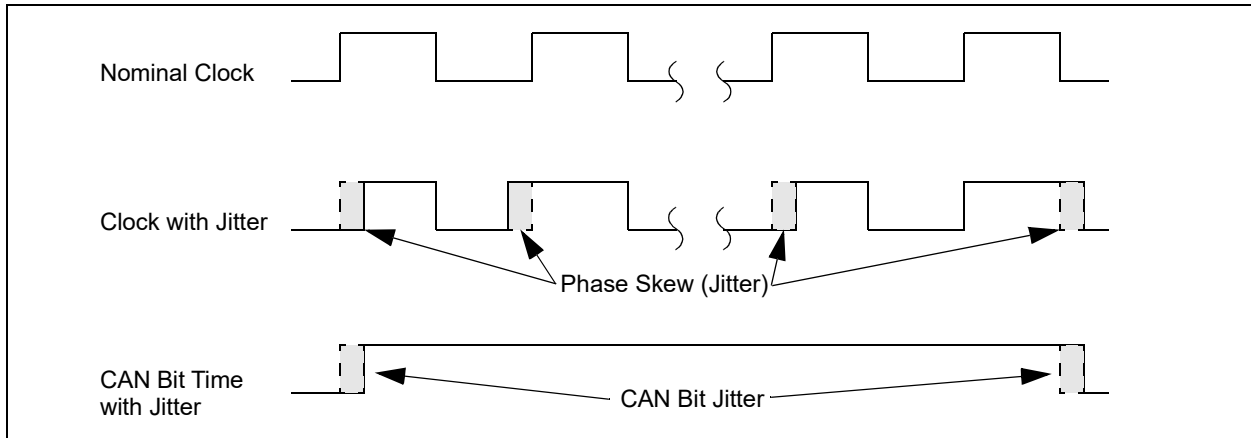
### 34.8.1 EXTERNAL CLOCK, INTERNAL CLOCK AND MEASURABLE JITTER IN HS-PLL BASED OSCILLATORS

The microcontroller clock frequency generated from a PLL circuit is subject to a jitter, also defined as Phase Jitter or Phase Skew. For its PIC18 Enhanced microcontrollers, Microchip specifies phase jitter ( $P_{jitter}$ ) as being 2% (Gaussian distribution, within three standard deviations, see Parameter PLL04 in Table 45-9) and Total Jitter ( $T_{jitter}$ ) as being  $2 * P_{jitter}$ .

The CAN protocol uses a bit-stuffing technique that inserts a bit of a given polarity following five bits with the opposite polarity. This gives a total of ten bits transmitted without resynchronization (compensation for jitter or phase error).

Given the random nature of the added jitter error, it can be shown that the total error caused by the jitter tends to cancel itself over time. For a period of ten bits, it is necessary to add only two jitter intervals to correct for jitter induced error: one interval in the beginning of the 10-bit period and another at the end. The overall effect is shown in Figure 34-5.

**FIGURE 34-5: EFFECTS OF PHASE JITTER ON THE MICROCONTROLLER CLOCK AND CAN BIT TIME**



Once these considerations are taken into account, it is possible to show that the relation between the jitter and the total frequency error can be defined as:

**EQUATION 34-4: JITTER AND TOTAL FREQUENCY ERROR**

$$\Delta f = \frac{T_{\text{jitter}}}{10 \times \text{NBT}} = \frac{2 \times P_{\text{jitter}}}{10 \times \text{NBT}}$$

where jitter is expressed in terms of time and NBT is the Nominal Bit Time.

For example, assume a CAN bit rate of 125 Kb/s, which gives an NBT of 8  $\mu$ s. For a 16 MHz clock generated from a 4x PLL, the jitter at this clock frequency is:

**EQUATION 34-5: 16 MHz CLOCK FROM 4x PLL JITTER:**

$$2\% \times \frac{1}{16 \text{ MHz}} = \frac{0.02}{16 \times 10^6} = 1.25 \text{ ns}$$

and resultant frequency error is:

**EQUATION 34-6: RESULTANT FREQUENCY ERROR:**

$$\frac{2 \times (1.25 \times 10^{-9})}{10 \times (8 \times 10^{-6})} = 3.125 \times 10^{-5} = 0.0031\%$$

Table 34-2 shows the relation between the clock generated by the PLL and the frequency error from jitter (measured jitter-induced error of 2%, Gaussian distribution, within three standard deviations), as a percentage of the nominal clock frequency.

This is clearly smaller than the expected drift of a crystal oscillator, typically specified at 100 ppm or 0.01%. If we add jitter to oscillator drift, we have a total frequency drift of 0.0132%. The total oscillator frequency errors for common clock frequencies and bit rates, including both drift and jitter, are shown in Table 34-3.

**TABLE 34-2: FREQUENCY ERROR FROM JITTER AT VARIOUS PLL GENERATED CLOCK SPEEDS**

| PLL Output | $P_{\text{jitter}}$ | $T_{\text{jitter}}$ | Frequency Error at Various Nominal Bit Times (Bit Rates) |                         |                         |                       |
|------------|---------------------|---------------------|----------------------------------------------------------|-------------------------|-------------------------|-----------------------|
|            |                     |                     | 8 $\mu$ s<br>(125 Kb/s)                                  | 4 $\mu$ s<br>(250 Kb/s) | 2 $\mu$ s<br>(500 Kb/s) | 1 $\mu$ s<br>(1 Mb/s) |
| 40 MHz     | 0.5 ns              | 1 ns                | 0.00125%                                                 | 0.00250%                | 0.005%                  | 0.01%                 |
| 24 MHz     | 0.83 ns             | 1.67 ns             | 0.00209%                                                 | 0.00418%                | 0.008%                  | 0.017%                |
| 16 MHz     | 1.25 ns             | 2.5 ns              | 0.00313%                                                 | 0.00625%                | 0.013%                  | 0.025%                |

**TABLE 34-3: TOTAL FREQUENCY ERROR AT VARIOUS PLL GENERATED CLOCK SPEEDS (100 PPM OSCILLATOR DRIFT, INCLUDING ERROR FROM JITTER)**

| Nominal PLL Output | Frequency Error at Various Nominal Bit Times (Bit Rates) |                         |                         |                       |
|--------------------|----------------------------------------------------------|-------------------------|-------------------------|-----------------------|
|                    | 8 $\mu$ s<br>(125 Kb/s)                                  | 4 $\mu$ s<br>(250 Kb/s) | 2 $\mu$ s<br>(500 Kb/s) | 1 $\mu$ s<br>(1 Mb/s) |
| 40 MHz             | 0.01125%                                                 | 0.01250%                | 0.015%                  | 0.02%                 |
| 24 MHz             | 0.01209%                                                 | 0.01418%                | 0.018%                  | 0.027%                |
| 16 MHz             | 0.01313%                                                 | 0.01625%                | 0.023%                  | 0.035%                |

## 34.8.2 TIME QUANTA

As already mentioned, the Time Quanta is a fixed unit derived from the oscillator period and baud rate prescaler. Its relationship to T<sub>BIT</sub> and the Nominal Bit Rate is shown in [Example 34-1](#).

### EXAMPLE 34-1: CALCULATING T<sub>Q</sub>, NOMINAL BIT RATE AND NOMINAL BIT TIME

$$T_Q (\mu s) = (2 * (BRP + 1)) / F_{OSC} (MHz)$$

$$T_{BIT} (\mu s) = T_Q (\mu s) * \text{number of } T_Q \text{ per bit interval}$$

$$\text{Nominal Bit Rate (bits/s)} = 1 / T_{BIT}$$

This frequency (F<sub>OSC</sub>) refers to the effective frequency used. If, for example, a 10 MHz external signal is used along with a PLL, then the effective frequency will be 4 x 10 MHz which equals 40 MHz.

#### CASE 1:

For F<sub>OSC</sub> = 16 MHz, BRP<5:0> = 00h and Nominal Bit Time = 8 T<sub>Q</sub>:

$$T_Q = (2 * 1) / 16 = 0.125 \mu s (125 \text{ ns})$$

$$T_{BIT} = 8 * 0.125 = 1 \mu s (10^{-6} s)$$

$$\text{Nominal Bit Rate} = 1 / 10^{-6} = 10^6 \text{ bits/s (1 Mb/s)}$$

#### CASE 2:

For F<sub>OSC</sub> = 20 MHz, BRP<5:0> = 01h and Nominal Bit Time = 8 T<sub>Q</sub>:

$$T_Q = (2 * 2) / 20 = 0.2 \mu s (200 \text{ ns})$$

$$T_{BIT} = 8 * 0.2 = 1.6 \mu s (1.6 * 10^{-6} s)$$

$$\text{Nominal Bit Rate} = 1 / 1.6 * 10^{-6} s = 625,000 \text{ bits/s (625 Kb/s)}$$

#### CASE 3:

For F<sub>OSC</sub> = 25 MHz, BRP<5:0> = 3Fh and Nominal Bit Time = 25 T<sub>Q</sub>:

is programmable from 4 to 25, the usable minimum is 8 T<sub>Q</sub>. There is no assurance that a bit time of less than 8 T<sub>Q</sub> in length will operate correctly.

## 34.8.3 SYNCHRONIZATION SEGMENT

This part of the bit time is used to synchronize the various CAN nodes on the bus. The edge of the input signal is expected to occur during the sync segment. The duration is 1 T<sub>Q</sub>.

## 34.8.4 PROPAGATION SEGMENT

This part of the bit time is used to compensate for physical delay times within the network. These delay times consist of the signal propagation time on the bus line and the internal delay time of the nodes. The length of the propagation segment can be programmed from 1 T<sub>Q</sub> to 8 T<sub>Q</sub> by setting the PRSEG<2:0> bits.

## 34.8.5 PHASE BUFFER SEGMENTS

The phase buffer segments are used to optimally locate the sampling point of the received bit within the Nominal Bit Time. The sampling point occurs between Phase Segment 1 and Phase Segment 2. These segments can be lengthened or shortened by the resynchronization process. The end of Phase Segment 1 determines the sampling point within a bit time. Phase Segment 1 is programmable from 1 T<sub>Q</sub> to 8 T<sub>Q</sub> in duration. Phase Segment 2 provides a delay before the next transmitted data transition and is also programmable from 1 T<sub>Q</sub> to 8 T<sub>Q</sub> in duration. However, due to IPT requirements, the actual minimum length of Phase Segment 2 is 2 T<sub>Q</sub>, or it may be defined to be equal to the greater of Phase Segment 1 or the Information Processing Time (IPT). The sampling point should be as late as possible or approximately 80% of the bit time.

## 34.8.6 SAMPLE POINT

The sample point is the point of time at which the bus level is read and the value of the received bit is determined. The sampling point occurs at the end of Phase Segment 1. If the bit timing is slow and contains many T<sub>Q</sub>, it is possible to specify multiple sampling of the bus line at the sample point. The value of the received bit is determined to be the value of the majority decision of three values. The three samples are taken at the sample point and twice before, with a time of T<sub>Q</sub>/2 between each sample.

The frequencies of the oscillators in the different nodes must be coordinated in order to provide a system wide specified Nominal Bit Time. This means that all oscillators must have a T<sub>OSC</sub> that is an integral divisor of T<sub>Q</sub>. It should also be noted that although the number of T<sub>Q</sub>

## 34.8.7 INFORMATION PROCESSING TIME

The Information Processing Time (IPT) is the time segment starting at the sample point that is reserved for calculation of the subsequent bit level. The CAN specification defines this time to be less than or equal to 2 T<sub>Q</sub>. The ECAN module defines this time to be 2 T<sub>Q</sub>. Thus, Phase Segment 2 must be at least 2 T<sub>Q</sub> long.

## 34.8.8 CLOCK SELECTION

The CLKSEL bit of the CIOCON register allows for selection between two CAN input clocks. When CLKSEL = 0 (default), the CAN clock (FOSC in the equations above) will be the same as the system clock. When CLKSEL = 1, the CAN clock will be the clock selected by the FEXTOSC Configuration bit, regardless of the system clock. This allows for the core of the device to be clocked by a PLL at 64 MHz (16 MHz HS crystal+4xPLL) while keeping the CAN clocked by the base 16 MHz HS crystal without the PLL, for example.

**Note:** If CLKSEL = 1, the system clock must be greater than or equal to the FEXTOSC selected clock. Having a slower system clock than the CAN clock will lead to unexpected behavior. The Information Processing Time (IPT) is the time segment starting at the sample point that is reserved for calculation of the subsequent bit level. The CAN specification defines this time to be less than or equal to 2 T<sub>Q</sub>. The CAN module defines this time to be 2 T<sub>Q</sub>. Thus, Phase Segment 2 must be at least 2 T<sub>Q</sub> long.

## 34.9 Synchronization

To compensate for phase shifts between the oscillator frequencies of each of the nodes on the bus, each CAN controller must be able to synchronize to the relevant signal edge of the incoming signal. When an edge in the transmitted data is detected, the logic will compare the location of the edge to the expected time (Sync\_Seg). The circuit will then adjust the values of Phase Segment 1 and Phase Segment 2 as necessary. There are two mechanisms used for synchronization.

### 34.9.1 HARD SYNCHRONIZATION

Hard synchronization is only done when there is a recessive to dominant edge during a bus Idle condition, indicating the start of a message. After hard synchronization, the bit time counters are restarted with Sync\_Seg. Hard synchronization forces the edge, which has occurred to lie within the synchronization segment of the restarted bit time. Due to the rules of synchronization, if a hard synchronization occurs, there will not be a resynchronization within that bit time.

### 34.9.2 RESYNCHRONIZATION

As a result of resynchronization, Phase Segment 1 may be lengthened or Phase Segment 2 may be shortened. The amount of lengthening or shortening of the phase buffer segments has an upper bound given by the Synchronization Jump Width (SJW). The value of the SJW will be added to Phase Segment 1 (see Figure 34-6) or subtracted from Phase Segment 2 (see Figure 34-7). The SJW is programmable between 1 T<sub>Q</sub> and 4 T<sub>Q</sub>.

Clocking information will only be derived from recessive to dominant transitions. The property, that only a fixed maximum number of successive bits have the same value, ensures resynchronization to the bit stream during a frame.

The phase error of an edge is given by the position of the edge relative to Sync\_Seg, measured in T<sub>Q</sub>. The phase error is defined in magnitude of T<sub>Q</sub> as follows:

- e = 0 if the edge lies within Sync\_Seg.
- e > 0 if the edge lies before the sample point.
- e < 0 if the edge lies after the sample point of the previous bit.

If the magnitude of the phase error is less than, or equal to, the programmed value of the Synchronization Jump Width, the effect of a resynchronization is the same as that of a hard synchronization.

If the magnitude of the phase error is larger than the Synchronization Jump Width and if the phase error is positive, then Phase Segment 1 is lengthened by an amount equal to the Synchronization Jump Width.

If the magnitude of the phase error is larger than the resynchronization jump width and if the phase error is negative, then Phase Segment 2 is shortened by an amount equal to the Synchronization Jump Width.

### 34.9.3 SYNCHRONIZATION RULES

- Only one synchronization within one bit time is allowed.
- An edge will be used for synchronization only if the value detected at the previous sample point (previously read bus value) differs from the bus value immediately after the edge.
- All other recessive to dominant edges fulfilling rules 1 and 2 will be used for resynchronization, with the exception that a node transmitting a dominant bit will not perform a resynchronization as a result of a recessive to dominant edge with a positive phase error.

**FIGURE 34-6: LENGTHENING A BIT PERIOD (ADDING SJW TO PHASE SEGMENT 1)**

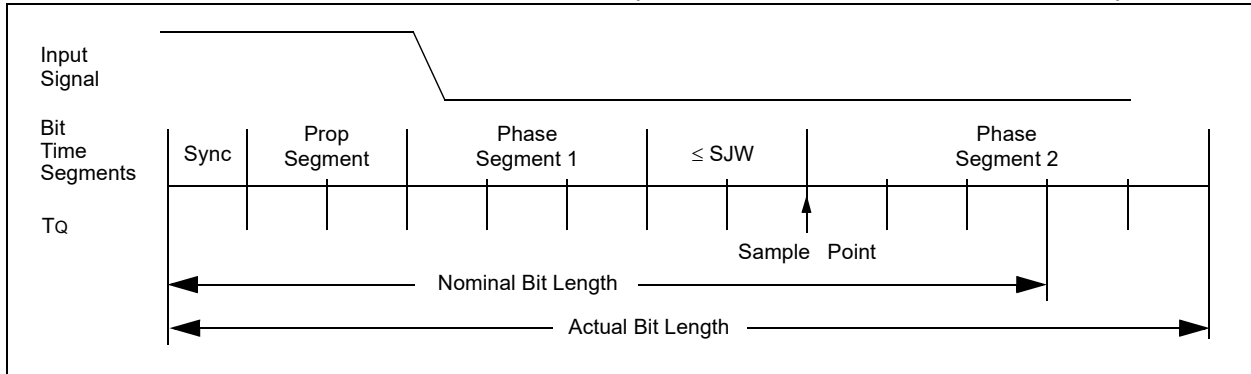
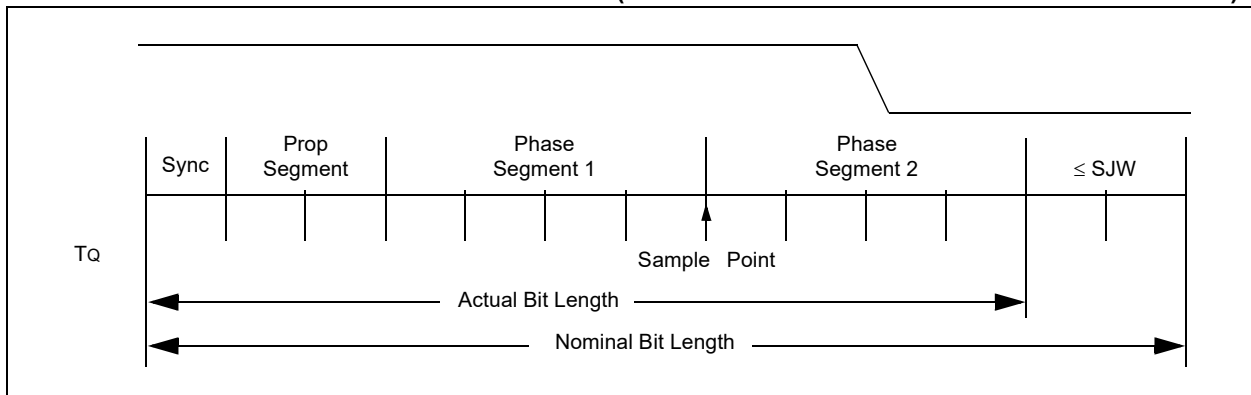


**FIGURE 34-7: SHORTENING A BIT PERIOD (SUBTRACTING SJW FROM PHASE SEGMENT 2)**



## 34.10 Programming Time Segments

Some requirements for programming of the time segments:

- $\text{Prop\_Seg} + \text{Phase\_Seg 1} \geq \text{Phase\_Seg 2}$
- $\text{Phase\_Seg 2} \geq \text{Sync Jump Width}$ .

For example, assume that a 125 kHz CAN baud rate is desired, using 20 MHz for  $F_{\text{osc}}$ . With a  $T_{\text{osc}}$  of 50 ns, a baud rate prescaler value of 04h gives a  $T_Q$  of 500 ns. To obtain a Nominal Bit Rate of 125 kHz, the Nominal Bit Time must be 8  $\mu\text{s}$  or 16  $T_Q$ .

Using 1  $T_Q$  for the Sync\_Seg, 2  $T_Q$  for the Prop\_Seg and 7  $T_Q$  for Phase Segment 1 would place the sample point at 10  $T_Q$  after the transition. This leaves 6  $T_Q$  for Phase Segment 2.

By the rules above, the Sync Jump Width could be the maximum of 4  $T_Q$ . However, normally a large SJW is only necessary when the clock generation of the different nodes is inaccurate or unstable, such as using ceramic resonators. Typically, an SJW of 1 is enough.

## 34.11 Oscillator Tolerance

As a rule of thumb, the bit timing requirements allow ceramic resonators to be used in applications with transmission rates of up to 125 Kbit/sec. For the full bus speed range of the CAN protocol, a quartz oscillator is required. Refer to ISO11898-1 for oscillator tolerance requirements.

## 34.12 Bit Timing Configuration Registers

The Baud Rate Control registers (BRGCON1, BRGCON2, BRGCON3) control the bit timing for the CAN bus interface. These registers can only be modified when the CAN module is in Configuration mode.

### 34.12.1 BRGCON1

The BRP bits control the baud rate prescaler. The  $\text{SJW} \langle 1:0 \rangle$  bits select the synchronization jump width in terms of multiples of  $T_Q$ .

## 34.12.2 BRGCON2

The PRSEG bits set the length of the propagation segment in terms of  $T_Q$ . The SEG1PH bits set the length of Phase Segment 1 in  $T_Q$ . The SAM bit controls how many times the RXCAN pin is sampled. Setting this bit to a '1' causes the bus to be sampled three times: twice at  $T_Q/2$  before the sample point and once at the normal sample point (which is at the end of Phase Segment 1). The value of the bus is determined to be the value read during at least two of the samples. If the SAM bit is set to a '0', then the RXCAN pin is sampled only once at the sample point. The SEG2PHTS bit controls how the length of Phase Segment 2 is determined. If this bit is set to a '1', then the length of Phase Segment 2 is determined by the SEG2PH bits of BRGCON3. If the SEG2PHTS bit is set to a '0', then the length of Phase Segment 2 is the greater of Phase Segment 1 and the information processing time (which is fixed at  $2 T_Q$  for the ECAN module).

## 34.12.3 BRGCON3

The PHSEG2<2:0> bits set the length (in  $T_Q$ ) of Phase Segment 2 if the SEG2PHTS bit is set to a '1'. If the SEG2PHTS bit is set to a '0', then the PHSEG2<2:0> bits have no effect.

## 34.13 Error Detection

The CAN protocol provides sophisticated error detection mechanisms. The following errors can be detected.

### 34.13.1 CRC ERROR

With the Cyclic Redundancy Check (CRC), the transmitter calculates special check bits for the bit sequence, from the start of a frame until the end of the data field. This CRC sequence is transmitted in the CRC field. The receiving node also calculates the CRC sequence using the same formula and performs a comparison to the received sequence. If a mismatch is detected, a CRC error has occurred and an error frame is generated. The message is repeated.

### 34.13.2 ACKNOWLEDGE ERROR

In the Acknowledge field of a message, the transmitter checks if the Acknowledge slot (which was sent out as a recessive bit) contains a dominant bit. If not, no other node has received the frame correctly. An Acknowledge error has occurred, an error frame is generated and the message will have to be repeated.

### 34.13.3 FORM ERROR

If a node detects a dominant bit in one of the four segments, including End-of-Frame (EOF), interframe space, Acknowledge delimiter or CRC delimiter, then a form error has occurred and an error frame is generated. The message is repeated.

### 34.13.4 BIT ERROR

A bit error occurs if a transmitter sends a dominant bit and detects a recessive bit, or if it sends a recessive bit and detects a dominant bit, when monitoring the actual bus level and comparing it to the just transmitted bit. In the case where the transmitter sends a recessive bit and a dominant bit is detected during the arbitration field and the Acknowledge slot, no bit error is generated because normal arbitration is occurring.

### 34.13.5 STUFF BIT ERROR

If, between the Start-of-Frame (SOF) and the CRC delimiter, six consecutive bits with the same polarity are detected, the bit stuffing rule has been violated. A stuff bit error occurs and an error frame is generated. The message is repeated.

### 34.13.6 ERROR STATES

Detected errors are made public to all other nodes via error frames. The transmission of the erroneous message is aborted and the frame is repeated as soon as possible. Furthermore, each CAN node is in one of the three error states; "error-active", "error-passive" or "bus-off", according to the value of the internal error counters. The error-active state is the usual state where the bus node can transmit messages and activate error frames (made of dominant bits) without any restrictions. In the error-passive state, messages and passive error frames (made of recessive bits) may be transmitted. The bus-off state makes it temporarily impossible for the node to participate in the bus communication. During this state, messages can neither be received nor transmitted.

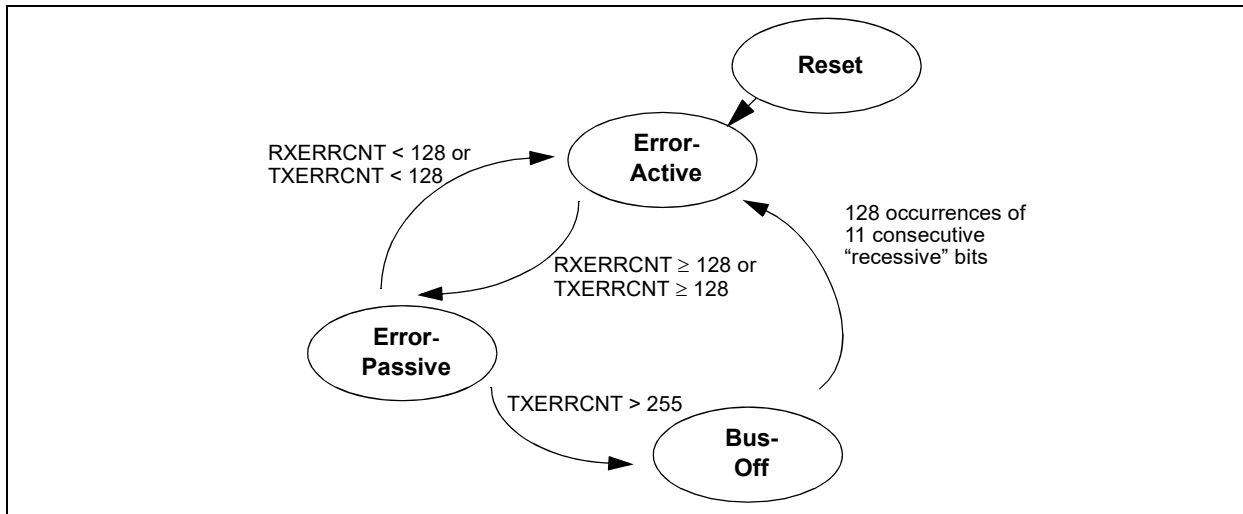
### 34.13.7 ERROR MODES AND ERROR COUNTERS

The CAN module contains two error counters: the Receive Error Counter (RXERRCNT) and the Transmit Error Counter (TXERRCNT). The values of both counters can be read by the MCU. These counters are incremented or decremented in accordance with the CAN bus specification.

The CAN module is error-active if both error counters are below the error-passive limit of 128. They are error-passive if at least one of the error counters equals or exceeds 128. They go to bus-off if the transmit error counter equals or exceeds the bus-off limit of 256. The devices remain in this state until the bus-off recovery sequence is finished. The bus-off recovery sequence consists of 128 occurrences of 11 consecutive recessive bits (see [Figure 34-8](#)). Note that the CAN module, after going bus-off, will recover back to error-active without any intervention by the MCU if the bus remains Idle for  $128 \times 11$  bit times. If this is not desired, the error Interrupt Service Routine should address this. The current Error mode of the CAN module can be read by the MCU via the COMSTAT register.

Additionally, there is an Error State Warning flag bit, EWARN, which is set if at least one of the error counters equals or exceeds the error warning limit of 96. EWARN is reset if both error counters are less than the error warning limit.

**FIGURE 34-8: ERROR MODES STATE DIAGRAM**





## 34.14 CAN Interrupts

The module has several sources of interrupts. Each of these interrupts can be individually enabled or disabled. The PIR5 register contains interrupt flags. The PIE5 register contains the enables for the eight main interrupts. A special set of read-only bits in the CANSTAT register, the ICODE bits, can be used in combination with a jump table for efficient handling of interrupts.

All interrupts have one source, with the exception of the error interrupt and buffer interrupts in Mode 1 and 2. Any of the error interrupt sources can set the error interrupt flag. The source of the error interrupt can be determined by reading the Communication Status register, COMSTAT. In Mode 1 and 2, there are two interrupt enable/disable and flag bits – one for all transmit buffers and the other for all receive buffers.

The interrupts can be broken up into two categories: receive and transmit interrupts.

The receive related interrupts are:

- Receive Interrupts
- Wake-up Interrupt
- Receiver Overrun Interrupt
- Receiver Warning Interrupt
- Receiver Error-Passive Interrupt

The transmit related interrupts are:

- Transmit Interrupts
- Transmitter Warning Interrupt
- Transmitter Error-Passive Interrupt
- Bus-Off Interrupt

### 34.14.1 INTERRUPT CODE BITS

To simplify the interrupt handling process in user firmware, the ECAN module encodes a special set of bits. In Mode 0, these bits are ICODE<3:1> in the CANSTAT register. In Mode 1 and 2, these bits are EICODE<4:0> in the CANSTAT register. Interrupts are internally prioritized such that the higher priority interrupts are assigned lower values. Once the highest priority interrupt condition has been cleared, the code for the next highest priority interrupt that is pending (if any) will be reflected by the ICODE bits (see [Table 34-4](#)). Note that only those interrupt sources that have their associated interrupt enable bit set will be reflected in the ICODE bits.

In Mode 2, when a receive message interrupt occurs, the EICODE bits will always consist of '10000'. User firmware may use FIFO Pointer bits to actually access the next available buffer.

### 34.14.2 TRANSMIT INTERRUPT

When the transmit interrupt is enabled, an interrupt will be generated when the associated transmit buffer becomes empty and is ready to be loaded with a new message. In Mode 0, there are separate interrupt enable/disable and flag bits for each of the three dedicated transmit buffers. The TXBnIF bit will be set to indicate the source of the interrupt. The interrupt is cleared by the MCU, resetting the TXBnIF bit to a '0'. In Mode 1 and 2, all transmit buffers share one interrupt enable/disable bit and one flag bit. In Mode 1 and 2, TXBnIE in PIE5 and TXBnIF in PIR5 indicate when a transmit buffer has completed transmission of its message. TXBnIF, TXBnIE and TXBnIP in PIR5, PIE5 and IPR5, respectively, are not used in Mode 1 and 2. Individual transmit buffer interrupts can be enabled or disabled by setting or clearing TXBnIE and B0IE register bits. When a shared interrupt occurs, user firmware must poll the TXREQ bit of all transmit buffers to detect the source of interrupt.

### 34.14.3 RECEIVE INTERRUPT

When the receive interrupt is enabled, an interrupt will be generated when a message has been successfully received and loaded into the associated receive buffer. This interrupt is activated immediately after receiving the End-of-Frame (EOF) field.

In Mode 0, the RXBnIF bit is set to indicate the source of the interrupt. The interrupt is cleared by the MCU, resetting the RXBnIF bit to a '0'.

In Mode 1 and 2, all receive buffers share RXBnIE, RXBnIF and RXBnIP in PIE5, PIR5 and IPR5, respectively. Individual receive buffer interrupts can be controlled by the TXBnIE and BIE0 registers. In Mode 1, when a shared receive interrupt occurs, user firmware must poll the RXFUL bit of each receive buffer to detect the source of interrupt. In Mode 2, a receive interrupt indicates that the new message is loaded into FIFO. FIFO can be read by using FIFO Pointer bits, FP.

**TABLE 34-4: VALUES FOR ICODE<2:0>**

| ICODE<br><2:0> | Interrupt            | Boolean Expression                                                                                                                                                                              |
|----------------|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 000            | None                 | $\overline{\text{ERR}} \cdot \overline{\text{WAK}} \cdot \overline{\text{TX0}} \cdot \overline{\text{TX1}} \cdot \overline{\text{TX2}} \cdot \overline{\text{RX0}} \cdot \overline{\text{RX1}}$ |
| 001            | Error                | ERR                                                                                                                                                                                             |
| 010            | TXB2                 | $\overline{\text{ERR}} \cdot \overline{\text{TX0}} \cdot \overline{\text{TX1}} \cdot \text{TX2}$                                                                                                |
| 011            | TXB1                 | $\overline{\text{ERR}} \cdot \overline{\text{TX0}} \cdot \text{TX1}$                                                                                                                            |
| 100            | TXB0                 | $\overline{\text{ERR}} \cdot \text{TX0}$                                                                                                                                                        |
| 101            | RXB1                 | $\overline{\text{ERR}} \cdot \overline{\text{TX0}} \cdot \overline{\text{TX1}} \cdot \overline{\text{TX2}} \cdot \overline{\text{RX0}} \cdot \text{RX1}$                                        |
| 110            | RXB0                 | $\overline{\text{ERR}} \cdot \overline{\text{TX0}} \cdot \overline{\text{TX1}} \cdot \overline{\text{TX2}} \cdot \text{RX0}$                                                                    |
| 111            | Wake on<br>Interrupt | $\overline{\text{ERR}} \cdot \overline{\text{TX0}} \cdot \overline{\text{TX1}} \cdot \overline{\text{TX2}} \cdot \overline{\text{RX0}} \cdot \overline{\text{RX1}} \cdot \text{WAK}$            |

**Legend:**

ERR = ERRIF \* ERRIE      RX0 = RXB0IF \* RXB0IE  
 TX0 = TXB0IF \* TXB0IE    RX1 = RXB1IF \* RXB1IE  
 TX1 = TXB1IF \* TXB1IE    WAK = WAKIF \* WAKIE  
 TX2 = TXB2IF \* TXB2IE

## 34.14.4 MESSAGE ERROR INTERRUPT

When an error occurs during transmission or reception of a message, the message error flag, IRIXIF, will be set and if the IRIXIE bit is set, an interrupt will be generated. This is intended to be used to facilitate baud rate determination when used in conjunction with Listen Only mode.

## 34.14.5 BUS ACTIVITY WAKE-UP INTERRUPT

When the ECAN module is in Sleep mode and the bus activity wake-up interrupt is enabled, an interrupt will be generated and the WAKIF bit will be set when activity is detected on the CAN bus. This interrupt causes the MCU to exit Sleep mode. The interrupt is reset by the MCU, clearing the WAKIF bit.

## 34.14.6 ERROR INTERRUPT

When the CAN module error interrupt (ERRIE in PIE5) is enabled, an interrupt is generated if an overflow condition occurs, or if the error state of the transmitter or receiver has changed. The error flags in COMSTAT will indicate one of the following conditions.

### 34.14.6.1 Receiver Overflow

An overflow condition occurs when the MAB has assembled a valid received message (the message meets the criteria of the acceptance filters) and the receive buffer associated with the filter is not available for loading of a new message. The associated RXBnOVFL bit in the COMSTAT register will be set to indicate the overflow condition. This bit must be cleared by the MCU. In mode 0, RXB0 and RXB1 have separate overflow bits. In modes 1 and 2, there is one shared bit that indicates a receive buffer has overflowed, but each buffer must be checked individually.

### 34.14.6.2 Receiver Warning

The receive error counter has reached the MCU warning limit of 96. This is indicated by the RXWARN bit of the COMSTAT register.

### 34.14.6.3 Transmitter Warning

The transmit error counter has reached the MCU warning limit of 96. This is indicated by the TXWARN bit of the COMSTAT register.

### 34.14.6.4 Receiver Bus Passive

This will occur when the device has gone to the error-passive state because the receive error counter is greater or equal to 128. This is indicated by the RXBP bit of the COMSTAT register.

### 34.14.6.5 Transmitter Bus Passive

This will occur when the device has gone to the error-passive state because the transmit error counter is greater or equal to 128. This is indicated by the TXBP bit of the COMSTAT register.

### 34.14.6.6 Bus-Off

The transmit error counter has exceeded 255 and the device has gone to bus-off state. This is indicated by the TXBO bit of the COMSTAT register.

## 34.15 CAN Module Registers

|                                                                      |
|----------------------------------------------------------------------|
| <b>Note:</b> Not all CAN registers are available in the Access Bank. |
|----------------------------------------------------------------------|

There are many control and data registers associated with the CAN module. For convenience, their descriptions have been grouped into the following sections:

- Control and Status Registers
- Dedicated Transmit Buffer Registers
- Dedicated Receive Buffer Registers
- Programmable TX/RX and Auto RTR Buffers
- Baud Rate Control Registers
- I/O Control Register
- Interrupt Status and Control Registers

Detailed descriptions of each register and their usage are described in the following sections.

### 34.15.1 CAN CONTROL AND STATUS REGISTERS

The registers described in this section control the overall operation of the CAN module and show its operational status.

## REGISTER 34-1: CANCON: CAN CONTROL REGISTER

|        |        |        |        |       |       |       |       |     |
|--------|--------|--------|--------|-------|-------|-------|-------|-----|
| Mode 0 | R/W-1  | R/W-0  | R/W-0  | R/S-0 | R/W-0 | R/W-0 | R/W-0 | U-0 |
|        | REQOP2 | REQOP1 | REQOP0 | ABAT  | WIN2  | WIN1  | WIN0  | —   |
| Mode 1 | R/W-1  | R/W-0  | R/W-0  | R/S-0 | U0    | U-0   | U-0   | U-0 |
|        | REQOP2 | REQOP1 | REQOP0 | ABAT  | —     | —     | —     | —   |
| Mode 2 | R/W-1  | R/W-0  | R/W-0  | R/S-0 | R-0   | R-0   | R-0   | R-0 |
|        | REQOP2 | REQOP1 | REQOP0 | ABAT  | FP3   | FP2   | FP1   | FP0 |
| bit 7  |        |        |        |       | bit 0 |       |       |     |

|                   |                  |                                    |
|-------------------|------------------|------------------------------------|
| <b>Legend:</b>    | S = Settable bit | U = Unimplemented bit, read as '0' |
| R = Readable bit  | W = Writable bit | '0' = Bit is cleared               |
| -n = Value at POR | '1' = Bit is set | x = Bit is unknown                 |

bit 7-5 **REQOP<2:0>:** Request CAN Operation Mode bits

1xx = Requests Configuration mode  
011 = Requests Listen Only mode  
010 = Requests Loopback mode  
001 = Disabled/Sleep mode  
000 = Requests Normal mode

bit 4 **ABAT:** Abort All Pending Transmissions bit

1 = Abort all pending transmissions (in all transmit buffers)<sup>(1)</sup>  
0 = Transmissions proceeding as normal

bit 3-1 **Mode 0:**

**WIN<2:0>:** Window Address bits

These bits select which of the CAN buffers to switch into the Access Bank area. This allows access to the buffer registers from any data memory bank. After a frame has caused an interrupt, the ICODE<3:0> bits can be copied to the WIN<2:0> bits to select the correct buffer. See [Example 34-2](#) for a code example.

111 = Receive Buffer 0  
110 = Receive Buffer 0  
101 = Receive Buffer 1  
100 = Transmit Buffer 0  
011 = Transmit Buffer 1  
010 = Transmit Buffer 2  
001 = Receive Buffer 0  
000 = Receive Buffer 0

bit 0 **Mode 0:**

**Unimplemented:** Read as '0'

bit 4-0 **Mode 1:**

**Unimplemented:** Read as '0'

**Mode 2:**

**FP<3:0>:** FIFO Read Pointer bits

These bits point to the message buffer to be read.

0000 = Receive Message Buffer 0  
0001 = Receive Message Buffer 1  
0010 = Receive Message Buffer 2  
0011 = Receive Message Buffer 3  
0100 = Receive Message Buffer 4  
0101 = Receive Message Buffer 5  
0110 = Receive Message Buffer 6  
0111 = Receive Message Buffer 7  
1000:1111 Reserved

**Note 1:** This bit will clear when all transmissions are aborted.

## REGISTER 34-2: CANSTAT: CAN STATUS REGISTER

| Mode 0      | R-1                    | R-0                    | R-0                    | R-0     | R-0     | R-0     | R-0     | U-0     |
|-------------|------------------------|------------------------|------------------------|---------|---------|---------|---------|---------|
|             | OPMODE2 <sup>(1)</sup> | OPMODE1 <sup>(1)</sup> | OPMODE0 <sup>(1)</sup> | —       | ICODE2  | ICODE1  | ICODE0  | —       |
| Mode 1,2    | R-1                    | R-0                    | R-0                    | R-0     | R-0     | R-0     | R-0     | R-0     |
|             | OPMODE2 <sup>(1)</sup> | OPMODE1 <sup>(1)</sup> | OPMODE0 <sup>(1)</sup> | EICODE4 | EICODE3 | EICODE2 | EICODE1 | EICODE0 |
| bit 7 bit 0 |                        |                        |                        |         |         |         |         |         |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-5 **OPMODE<2:0>**: Operation Mode Status bits<sup>(1)</sup>

111 = Reserved

110 = Reserved

101 = Reserved

100 = Configuration mode

011 = Listen Only mode

010 = Loopback mode

001 = Disable/Sleep mode

000 = Normal mode

bit 4 **Mode 0:**

**Unimplemented:** Read as '0'

bit 3-1,4-0 **Mode 0:**

**ICODE<2:0>**: Interrupt Code bits

When an interrupt occurs, a prioritized coded interrupt value will be present in these bits. This code indicates the source of the interrupt. By copying ICODE<3:1> to WIN<3:0> (Mode 0) or EICODE<4:0> to EWIN<4:0> (Mode 1 and 2), it is possible to select the correct buffer to map into the Access Bank area. See [Example 34-2](#) for a code example. To simplify the description, the following table lists all five bits.

|                         | Mode 0 | Mode 1 | Mode 2               |
|-------------------------|--------|--------|----------------------|
| No interrupt            | 00000  | 00000  | 00000                |
| CAN bus error interrupt | 00010  | 00010  | 00010                |
| TXB2 interrupt          | 00100  | 00100  | 00100                |
| TXB1 interrupt          | 00110  | 00110  | 00110                |
| TXB0 interrupt          | 01000  | 01000  | 01000                |
| RXB1 interrupt          | 01010  | 10001  | -----                |
| RXB0 interrupt          | 01100  | 10000  | 10000                |
| Wake-up interrupt       | 01110  | 01110  | 01110                |
| RXB0 interrupt          | -----  | 10000  | 10000                |
| RXB1 interrupt          | -----  | 10001  | 10000                |
| RX/TX B0 interrupt      | -----  | 10010  | 10010 <sup>(2)</sup> |
| RX/TX B1 interrupt      | -----  | 10011  | 10011 <sup>(2)</sup> |
| RX/TX B2 interrupt      | -----  | 10100  | 10100 <sup>(2)</sup> |
| RX/TX B3 interrupt      | -----  | 10101  | 10101 <sup>(2)</sup> |
| RX/TX B4 interrupt      | -----  | 10110  | 10110 <sup>(2)</sup> |
| RX/TX B5 interrupt      | -----  | 10111  | 10111 <sup>(2)</sup> |

bit 0 **Mode 0:**

**Unimplemented:** Read as '0'

bit 4-0 **Mode 1, 2:**

**EICODE<4:0>**: Interrupt Code bits

See ICODE<3:1> above.

**Note 1:** To achieve maximum power saving and/or able to wake-up on CAN bus activity, switch the CAN module in Disable/Sleep mode before putting the device to Sleep.

**2:** If the buffer is configured as a receiver, the EICODE bits will contain '10000' upon interrupt.

## EXAMPLE 34-2: CHANGING TO CONFIGURATION MODE

```

; Request Configuration mode.
MOVLW  B'10000000'          ; Set to Configuration Mode.
MOVWF  CANCON

; A request to switch to Configuration mode may not be immediately honored.
; Module will wait for CAN bus to be idle before switching to Configuration Mode.
; Request for other modes such as Loopback, Disable etc. may be honored immediately.
; It is always good practice to wait and verify before continuing.
ConfigWait:
MOVWF  CANSTAT, W           ; Read current mode state.
ANDLW  B'10000000'         ; Interested in OPMODE bits only.
TSTFSZ WREG                 ; Is it Configuration mode yet?
BRA    ConfigWait           ; No. Continue to wait...
; Module is in Configuration mode now.
; Modify configuration registers as required.
; Switch back to Normal mode to be able to communicate.

```

## EXAMPLE 34-3: WIN AND ICODE BITS USAGE IN INTERRUPT SERVICE ROUTINE TO ACCESS TX/RX BUFFERS

```

; Save application required context.
; Poll interrupt flags and determine source of interrupt
; This was found to be CAN interrupt
; TempCANCON and TempCANSTAT are variables defined in Access Bank low
MOVFF  CANCON, TempCANCON    ; Save CANCON.WIN bits
                                ; This is required to prevent CANCON
                                ; from corrupting CAN buffer access
                                ; in-progress while this interrupt
                                ; occurred
MOVFF  CANSTAT, TempCANSTAT  ; Save CANSTAT register
                                ; This is required to make sure that
                                ; we use same CANSTAT value rather
                                ; than one changed by another CAN
                                ; interrupt.
MOVWF  TempCANSTAT, W        ; Retrieve ICODE bits
ANDLW  B'00001110'
ADDWF  PCL, F                ; Perform computed GOTO
                                ; to corresponding interrupt cause
BRA    NoInterrupt           ; 000 = No interrupt
BRA    ErrorInterrupt        ; 001 = Error interrupt
BRA    TXB2Interrupt         ; 010 = TXB2 interrupt
BRA    TXB1Interrupt         ; 011 = TXB1 interrupt
BRA    TXB0Interrupt         ; 100 = TXB0 interrupt
BRA    RXB1Interrupt         ; 101 = RXB1 interrupt
BRA    RXB0Interrupt         ; 110 = RXB0 interrupt
                                ; 111 = Wake-up on interrupt

WakeupInterrupt
BCF    PIR3, WAKIF           ; Clear the interrupt flag
;
; User code to handle wake-up procedure
;
; Continue checking for other interrupt source or return from here
...
NoInterrupt
...
                                ; PC should never vector here. User may
                                ; place a trap such as infinite loop or pin/port
                                ; indication to catch this error.

```

## EXAMPLE 34-2: WIN AND ICODE BITS USAGE IN INTERRUPT SERVICE ROUTINE TO ACCESS TX/RX BUFFERS (CONTINUED)

```

ErrorInterrupt
    BCF    PIR3, ERRIF                ; Clear the interrupt flag
    ...                               ; Handle error.
    RETFIE

TXB2Interrupt
    BCF    PIR3, TXB2IF              ; Clear the interrupt flag
    GOTO   AccessBuffer

TXB1Interrupt
    BCF    PIR3, TXB1IF              ; Clear the interrupt flag
    GOTO   AccessBuffer

TXB0Interrupt
    BCF    PIR3, TXB0IF              ; Clear the interrupt flag
    GOTO   AccessBuffer

RXB1Interrupt
    BCF    PIR3, RXB1IF              ; Clear the interrupt flag
    GOTO   Accessbuffer

RXB0Interrupt
    BCF    PIR3, RXB0IF              ; Clear the interrupt flag
    GOTO   AccessBuffer

AccessBuffer
    ; This is either TX or RX interrupt
    ; Copy CANSTAT.ICODE bits to CANCON.WIN bits
    MOVF   TempCANCON, W              ; Clear CANCON.WIN bits before copying
    ; new ones.
    ANDLW  B'11110001'               ; Use previously saved CANCON value to
    ; make sure same value.
    MOVWF  TempCANCON                 ; Copy masked value back to TempCANCON
    MOVF   TempCANSTAT, W             ; Retrieve ICODE bits
    ANDLW  B'00001110'               ; Use previously saved CANSTAT value
    ; to make sure same value.
    IORWF  TempCANCON                 ; Copy ICODE bits to WIN bits.
    MOVFF  TempCANCON, CANCON         ; Copy the result to actual CANCON
    ; Access current buffer...
    ; User code
    ; Restore CANCON.WIN bits
    MOVF   CANCON, W                  ; Preserve current non WIN bits
    ANDLW  B'11110001'               ; Restore original WIN bits
    IORWF  TempCANCON
    ; Do not need to restore CANSTAT - it is read-only register.
    ; Return from interrupt or check for another module interrupt source

```

## REGISTER 34-3: ECANCON: ENHANCED CAN CONTROL REGISTER

| R/W-0                 | R/W-0                 | R/W-0                 | R/W-1 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|-----------------------|-----------------------|-----------------------|-------|-------|-------|-------|-------|
| MDSEL1 <sup>(1)</sup> | MDSEL0 <sup>(1)</sup> | FIFOWM <sup>(2)</sup> | EWIN4 | EWIN3 | EWIN2 | EWIN1 | EWIN0 |
| bit 7                 |                       |                       |       |       |       |       | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-6 **MDSEL<1:0>:** Mode Select bits<sup>(1)</sup>

- 00 = Legacy mode (Mode 0, default)
- 01 = Enhanced Legacy mode (Mode 1)
- 10 = Enhanced FIFO mode (Mode 2)
- 11 = Reserved

bit 5 **FIFOWM:** FIFO High Water Mark bit<sup>(2)</sup>

- 1 = Will cause FIFO interrupt when one receive buffer remains
- 0 = Will cause FIFO interrupt when four receive buffers remain<sup>(3)</sup>

bit 4-0 **EWIN<4:0>:** Enhanced Window Address bits

These bits map the group of 16 banked CAN SFRs into Access Bank addresses, 0F60-0F6Dh. The exact group of registers to map is determined by the binary value of these bits.

#### Mode 0:

**Unimplemented:** Read as '0'

#### Mode 1, 2:

- 00000 = Acceptance Filters 0, 1, 2 and BRGCON2, 3
- 00001 = Acceptance Filters 3, 4, 5 and BRGCON1, CIOCON
- 00010 = Acceptance Filter Masks, Error and Interrupt Control
- 00011 = Transmit Buffer 0
- 00100 = Transmit Buffer 1
- 00101 = Transmit Buffer 2
- 00110 = Acceptance Filters 6, 7, 8
- 00111 = Acceptance Filters 9, 10, 11
- 01000 = Acceptance Filters 12, 13, 14
- 01001 = Acceptance Filter 15
- 01010-01110 = Reserved
- 01111 = RXINT0, RXINT1
- 10000 = Receive Buffer 0
- 10001 = Receive Buffer 1
- 10010 = TX/RX Buffer 0
- 10011 = TX/RX Buffer 1
- 10100 = TX/RX Buffer 2
- 10101 = TX/RX Buffer 3
- 10110 = TX/RX Buffer 4
- 10111 = TX/RX Buffer 5
- 11000-11111 = Reserved

**Note 1:** These bits can only be changed in Configuration mode. See [Register 34-1](#) to change to Configuration mode.

**2:** This bit is used in Mode 2 only.

**3:** If FIFO is configured to contain four or less buffers, then the FIFO interrupt will trigger.



## REGISTER 34-4: COMSTAT: COMMUNICATION STATUS REGISTER

|                                                |           |          |      |      |      |        |        |       |
|------------------------------------------------|-----------|----------|------|------|------|--------|--------|-------|
| Mode 0                                         | R/C-0     | R/C-0    | R-0  | R-0  | R-0  | R-0    | R-0    | R-0   |
|                                                | RXB0OVFL  | RXB1OVFL | TXBO | TXBP | RXBP | TXWARN | RXWARN | EWARN |
| Mode 1                                         | R/C-0     | R/C-0    | R-0  | R-0  | R-0  | R-0    | R-0    | R-0   |
|                                                | —         | RXBnOVFL | TXB0 | TXBP | RXBP | TXWARN | RXWARN | EWARN |
| Mode 2                                         | R/C-0     | R/C-0    | R-0  | R-0  | R-0  | R-0    | R-0    | R-0   |
|                                                | FIFOEMPTY | RXBnOVFL | TXBO | TXBP | RXBP | TXWARN | RXWARN | EWARN |
| bit 7 <span style="float: right;">bit 0</span> |           |          |      |      |      |        |        |       |

|                   |                                    |
|-------------------|------------------------------------|
| <b>Legend:</b>    | C = Clearable bit                  |
| R = Readable bit  | W = Writable bit                   |
| -n = Value at POR | '1' = Bit is set                   |
|                   | U = Unimplemented bit, read as '0' |
|                   | '0' = Bit is cleared               |
|                   | x = Bit is unknown                 |

- bit 7 **Mode 0:**  
**RXB0OVFL:** Receive Buffer 0 Overflow bit  
 1 = Receive Buffer 0 has overflowed  
 0 = Receive Buffer 0 has not overflowed  
**Mode 1:**  
**Unimplemented:** Read as '0'  
**Mode 2:**  
**FIFOEMPTY:** FIFO Not Empty bit  
 1 = Receive FIFO is not empty  
 0 = Receive FIFO is empty
- bit 6 **Mode 0:**  
**RXB1OVFL:** Receive Buffer 1 Overflow bit  
 1 = Receive Buffer 1 has overflowed  
 0 = Receive Buffer 1 has not overflowed  
**Mode 1, 2:**  
**RXBnOVFL:** Receive Buffer n Overflow bit  
 1 = Receive Buffer n has overflowed  
 0 = Receive Buffer n has not overflowed
- bit 5 **TXBO:** Transmitter Bus-Off bit  
 1 = Transmit error counter > 255  
 0 = Transmit error counter ≤ 255
- bit 4 **TXBP:** Transmitter Bus Passive bit  
 1 = Transmit error counter > 127  
 0 = Transmit error counter ≤ 127
- bit 3 **RXBP:** Receiver Bus Passive bit  
 1 = Receive error counter > 127  
 0 = Receive error counter ≤ 127
- bit 2 **TXWARN:** Transmitter Warning bit  
 1 = Transmit error counter > 95  
 0 = Transmit error counter ≤ 95
- bit 1 **RXWARN:** Receiver Warning bit  
 1 = 127 ≥ Receive error counter > 95  
 0 = Receive error counter ≤ 95
- bit 0 **EWARN:** Error Warning bit  
 This bit is a flag of the RXWARN and TXWARN bits.  
 1 = The RXWARN or the TXWARN bits are set  
 0 = Neither the RXWARN or the TXWARN bits are set

## 34.15.2 DEDICATED CAN TRANSMIT BUFFER REGISTERS

This section describes the dedicated CAN Transmit Buffer registers and their associated control registers.

### REGISTER 34-5: TXBnCON: TRANSMIT BUFFER n CONTROL REGISTERS [0 ≤ n ≤ 2]

| RC-0  | R-0                  | R-0                   | R-0                  | R/W-0                | U-0 | R/W-0                 | R/W-0                 |
|-------|----------------------|-----------------------|----------------------|----------------------|-----|-----------------------|-----------------------|
| TXBIF | TXABT <sup>(1)</sup> | TXLARB <sup>(1)</sup> | TXERR <sup>(1)</sup> | TXREQ <sup>(2)</sup> | —   | TXPRI1 <sup>(3)</sup> | TXPRI0 <sup>(3)</sup> |
| bit 7 |                      |                       |                      |                      |     |                       | bit 0                 |

|                   |                                    |
|-------------------|------------------------------------|
| <b>Legend:</b>    | C = Clearable bit                  |
| R = Readable bit  | W = Writable bit                   |
| -n = Value at POR | '1' = Bit is set                   |
|                   | U = Unimplemented bit, read as '0' |
|                   | '0' = Bit is cleared               |
|                   | x = Bit is unknown                 |

- bit 7 **TXBIF:** Transmit Buffer Interrupt Flag bit  
 1 = Transmit buffer has completed transmission of a message and may be reloaded  
 0 = Transmit buffer has not completed transmission of a message
- bit 6 **TXABT:** Transmission Aborted Status bit<sup>(1)</sup>  
 1 = Message was aborted  
 0 = Message was not aborted
- bit 5 **TXLARB:** Transmission Lost Arbitration Status bit<sup>(1)</sup>  
 1 = Message lost arbitration while being sent  
 0 = Message did not lose arbitration while being sent
- bit 4 **TXERR:** Transmission Error Detected Status bit<sup>(1)</sup>  
 1 = A bus error occurred while the message was being sent  
 0 = A bus error did not occur while the message was being sent
- bit 3 **TXREQ:** Transmit Request Status bit<sup>(2)</sup>  
 1 = Requests sending a message; clears the TXABT, TXLARB and TXERR bits  
 0 = Automatically cleared when the message is successfully sent
- bit 2 **Unimplemented:** Read as '0'
- bit 1-0 **TXPRI<1:0>:** Transmit Priority bits<sup>(3)</sup>  
 11 = Priority Level 3 (highest priority)  
 10 = Priority Level 2  
 01 = Priority Level 1  
 00 = Priority Level 0 (lowest priority)

- Note 1:** This bit is automatically cleared when TXREQ is set.
- 2:** While TXREQ is set, Transmit Buffer registers remain read-only. Clearing this bit in software while the bit is set will request a message abort.
- 3:** These bits define the order in which transmit buffers will be transferred. They do not alter the CAN message identifier.

## REGISTER 34-6: TXBnSIDH: TRANSMIT BUFFER 'n' STANDARD IDENTIFIER REGISTERS, HIGH BYTE [0 ≤ n ≤ 2]

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|-------|-------|-------|-------|-------|-------|-------|-------|
| SID10 | SID9  | SID8  | SID7  | SID6  | SID5  | SID4  | SID3  |
| bit 7 |       |       |       |       |       |       | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-0

**SID<10:3>**: Standard Identifier bits (if EXIDE (TXBnSIDL<3>) = 0)

Extended Identifier bits, EID<28:21> (if EXIDE = 1).

## REGISTER 34-7: TXBnSIDL: TRANSMIT BUFFER 'n' STANDARD IDENTIFIER REGISTERS, LOW BYTE [0 ≤ n ≤ 2]

| R/W-x | R/W-x | R/W-x | U-0 | R/W-x | U-0 | R/W-x | R/W-x |
|-------|-------|-------|-----|-------|-----|-------|-------|
| SID2  | SID1  | SID0  | —   | EXIDE | —   | EID17 | EID16 |
| bit 7 |       |       |     |       |     |       | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-5

**SID<2:0>**: Standard Identifier bits (if EXIDE (TXBnSIDL<3>) = 0)

Extended Identifier bits, EID<20:18> (if EXIDE = 1).

bit 4

**Unimplemented**: Read as '0'

bit 3

**EXIDE**: Extended Identifier Enable bit

1 = Message will transmit extended ID, SID<10:0> become EID<28:18>

0 = Message will transmit standard ID, EID<17:0> are ignored

bit 2

**Unimplemented**: Read as '0'

bit 1-0

**EID<17:16>**: Extended Identifier bits

## REGISTER 34-8: TXBnEIDH: TRANSMIT BUFFER 'n' EXTENDED IDENTIFIER REGISTERS, HIGH BYTE [0 ≤ n ≤ 2]

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|-------|-------|-------|-------|-------|-------|-------|-------|
| EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9  | EID8  |
| bit 7 |       |       |       |       |       |       | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-0

**EID<15:8>**: Extended Identifier bits (not used when transmitting standard identifier message)

## REGISTER 34-9: TXBnEIDL: TRANSMIT BUFFER 'n' EXTENDED IDENTIFIER REGISTERS, LOW BYTE [ $0 \leq n \leq 2$ ]

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|-------|-------|-------|-------|-------|-------|-------|-------|
| EID7  | EID6  | EID5  | EID4  | EID3  | EID2  | EID1  | EID0  |
| bit 7 |       |       |       |       |       |       | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-0

**EID<7:0>**: Extended Identifier bits (not used when transmitting standard identifier message)

## REGISTER 34-10: TXBnDm: TRANSMIT BUFFER 'n' DATA FIELD BYTE 'm' REGISTERS [ $0 \leq n \leq 2, 0 \leq m \leq 7$ ]

| R/W-x   | R/W-x   | R/W-x   | R/W-x   | R/W-x   | R/W-x   | R/W-x   | R/W-x   |
|---------|---------|---------|---------|---------|---------|---------|---------|
| TXBnDm7 | TXBnDm6 | TXBnDm5 | TXBnDm4 | TXBnDm3 | TXBnDm2 | TXBnDm1 | TXBnDm0 |
| bit 7   |         |         |         |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-0

**TXBnDm<7:0>**: Transmit Buffer n Data Field Byte m bits (where  $0 \leq n < 3$  and  $0 \leq m < 8$ )

Each transmit buffer has an array of registers. For example, Transmit Buffer 0 has 7 registers: TXB0D0 to TXB0D7.

## REGISTER 34-11: TXBnDLC: TRANSMIT BUFFER 'n' DATA LENGTH CODE REGISTERS [ $0 \leq n \leq 2$ ]

| U-0   | R/W-x | U-0 | U-0 | R/W-x | R/W-x | R/W-x | R/W-x |
|-------|-------|-----|-----|-------|-------|-------|-------|
| —     | TXRTR | —   | —   | DLC3  | DLC2  | DLC1  | DLC0  |
| bit 7 |       |     |     | bit 0 |       |       |       |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7 **Unimplemented:** Read as '0'

bit 6 **TXRTR:** Transmit Remote Frame Transmission Request bit

1 = Transmitted message will have the TXRTR bit set

0 = Transmitted message will have the TXRTR bit cleared

bit 5-4 **Unimplemented:** Read as '0'

bit 3-0 **DLC<3:0>:** Data Length Code bits

1111 = Reserved

1110 = Reserved

1101 = Reserved

1100 = Reserved

1011 = Reserved

1010 = Reserved

1001 = Reserved

1000 = Data length = 8 bytes

0111 = Data length = 7 bytes

0110 = Data length = 6 bytes

0101 = Data length = 5 bytes

0100 = Data length = 4 bytes

0011 = Data length = 3 bytes

0010 = Data length = 2 bytes

0001 = Data length = 1 bytes

0000 = Data length = 0 bytes

## REGISTER 34-12: TXERRCNT: TRANSMIT ERROR COUNT REGISTER

| R-0   | R-0  | R-0  | R-0  | R-0  | R-0  | R-0  | R-0   |
|-------|------|------|------|------|------|------|-------|
| TEC7  | TEC6 | TEC5 | TEC4 | TEC3 | TEC2 | TEC1 | TEC0  |
| bit 7 |      |      |      |      |      |      | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-0 **TEC<7:0>:** Transmit Error Counter bits

This register contains a value which is derived from the rate at which errors occur. When the error count overflows, the bus-off state occurs. When the bus has 128 occurrences of 11 consecutive recessive bits, the counter value is cleared.

## EXAMPLE 34-3: TRANSMITTING A CAN MESSAGE USING BANKED METHOD

```
; Need to transmit Standard Identifier message 123h using TXB0 buffer.
; To successfully transmit, CAN module must be either in Normal or Loopback mode.
; TXB0 buffer is not in access bank. And since we want banked method, we need to make sure
; that correct bank is selected.
BANKSEL TXB0CON                ; One BANKSEL in beginning will make sure that we are
                                ; in correct bank for rest of the buffer access.

; Now load transmit data into TXB0 buffer.
MOVLW MY_DATA_BYTE1            ; Load first data byte into buffer
MOVWF TXB0D0                    ; Compiler will automatically set "BANKED" bit
; Load rest of data bytes - up to 8 bytes into TXB0 buffer.
...
; Load message identifier
MOVLW 60H                       ; Load SID2:SID0, EXIDE = 0
MOVWF TXB0SIDL
MOVLW 24H                       ; Load SID10:SID3
MOVWF TXB0SIDH
; No need to load TXB0EIDL:TXB0EIDH, as we are transmitting Standard Identifier Message only.

; Now that all data bytes are loaded, mark it for transmission.
MOVLW B'00001000'              ; Normal priority; Request transmission
MOVWF TXB0CON

; If required, wait for message to get transmitted
BTFSC TXB0CON, TXREQ            ; Is it transmitted?
BRA $-2                        ; No. Continue to wait...

; Message is transmitted.
```

## EXAMPLE 34-4: TRANSMITTING A CAN MESSAGE USING WIN BITS

```
; Need to transmit Standard Identifier message 123h using TXB0 buffer.
; To successfully transmit, CAN module must be either in Normal or Loopback mode.
; TXB0 buffer is not in access bank. Use WIN bits to map it to RXB0 area.
MOVWF  CANCON, W                ; WIN bits are in lower 4 bits only. Read CANCON
                                   ; register to preserve all other bits. If operation
                                   ; mode is already known, there is no need to preserve
                                   ; other bits.

ANDLW  B'11110000'              ; Clear WIN bits.
IORLW  B'00001000'              ; Select Transmit Buffer 0
MOVWF  CANCON                   ; Apply the changes.

; Now TXB0 is mapped in place of RXB0. All future access to RXB0 registers will actually
; yield TXB0 register values.

; Load transmit data into TXB0 buffer.
MOVLW  MY_DATA_BYTE1            ; Load first data byte into buffer
MOVWF  RXBOD0                   ; Access TXBOD0 via RXBOD0 address.
; Load rest of the data bytes - up to 8 bytes into "TXB0" buffer using RXB0 registers.
...
; Load message identifier
MOVLW  60H                      ; Load SID2:SID0, EXIDE = 0
MOVWF  RXBOSIDL
MOVLW  24H                      ; Load SID10:SID3
MOVWF  RXBOSIDH
; No need to load RXB0EIDL:RXB0EIDH, as we are transmitting Standard Identifier Message only.

; Now that all data bytes are loaded, mark it for transmission.
MOVLW  B'00001000'              ; Normal priority; Request transmission
MOVWF  RXBOCON

; If required, wait for message to get transmitted
BTFSC  RXBOCON, TXREQ            ; Is it transmitted?
BRA    $-2                      ; No. Continue to wait...

; Message is transmitted.
; If required, reset the WIN bits to default state.
```

## 34.15.3 DEDICATED CAN RECEIVE BUFFER REGISTERS

This section shows the dedicated CAN Receive Buffer registers with their associated control registers.

### REGISTER 34-13: RXB0CON: RECEIVE BUFFER 0 CONTROL REGISTER

| Mode 0 | R/C-0                | R/W-0 | R/W-0 | U-0 | R-0     | R/W-0    | R-0                  | R-0     |
|--------|----------------------|-------|-------|-----|---------|----------|----------------------|---------|
|        | RXFUL <sup>(1)</sup> | RXM1  | RXM0  | —   | RXRTRRO | RXB0DBEN | JTOFF <sup>(2)</sup> | FILHIT0 |

| Mode 1,2                                       | R/C-0                | R/W-0 | R-0   | R-0      | R-0     | R-0     | R-0     | R-0     |
|------------------------------------------------|----------------------|-------|-------|----------|---------|---------|---------|---------|
|                                                | RXFUL <sup>(1)</sup> | RXM1  | RTRRO | FILHITF4 | FILHIT3 | FILHIT2 | FILHIT1 | FILHIT0 |
| bit 7 <span style="float: right;">bit 0</span> |                      |       |       |          |         |         |         |         |

|                   |                      |
|-------------------|----------------------|
| <b>Legend:</b>    | C = Clearable bit    |
| R = Readable bit  | W = Writable bit     |
| -n = Value at POR | '1' = Bit is set     |
|                   | '0' = Bit is cleared |
|                   | x = Bit is unknown   |

- bit 7 **RXFUL**: Receive Full Status bit<sup>(1)</sup>  
 1 = Receive buffer contains a received message  
 0 = Receive buffer is open to receive a new message
- bit 6,6-5 **Mode 0:**  
**RXM<1:0>**: Receive Buffer Mode bit 1 (combines with RXM0 to form RXM<1:0> bits, see bit 5)  
 11 = Receive all messages (including those with errors); filter criteria is ignored  
 10 = Receive only valid messages with extended identifier; EXIDEN in RXFnSIDL must be '1'  
 01 = Receive only valid messages with standard identifier; EXIDEN in RXFnSIDL must be '0'  
 00 = Receive all valid messages as per the EXIDEN bit in the RXFnSIDL register  
**Mode 1, 2:**  
**RXM1**: Receive Buffer Mode bit 1  
 1 = Receive all messages (including those with errors); acceptance filters are ignored  
 0 = Receive all valid messages as per acceptance filters
- bit 5 **Mode 0:**  
**RXM0**: Receive Buffer Mode bit 0 (combines with RXM1 to form RXM<1:0>bits, see bit 6)  
**Mode 1, 2:**  
**RTRRO**: Remote Transmission Request bit for Received Message (read-only)  
 1 = A remote transmission request is received  
 0 = A remote transmission request is not received
- bit 4 **Mode 0:**  
**Unimplemented**: Read as '0'  
**Mode 1, 2:**  
**FILHIT<4:0>**: Filter Hit bit 4  
 This bit combines with other bits to form filter acceptance bits<4:0>.
- bit 3 **Mode 0:**  
**RXRTRRO**: Remote Transmission Request bit for Received Message (read-only)  
 1 = A remote transmission request is received  
 0 = A remote transmission request is not received  
**Mode 1, 2:**  
**FILHIT<4:0>**: Filter Hit bit 3  
 This bit combines with other bits to form filter acceptance bits<4:0>.

**Note 1:** This bit is set by the CAN module upon receiving a message and must be cleared by software after the buffer is read. As long as RXFUL is set, no new message will be loaded and the buffer will be considered full. After clearing the RXFUL flag, the PIR5 bit, RXB0IF, can be cleared. If RXB0IF is cleared, but RXFUL is not cleared, then RXB0IF is set again.

**2:** This bit allows the same filter jump table for both RXB0CON and RXB1CON.



## REGISTER 34-13: RXB0CON: RECEIVE BUFFER 0 CONTROL REGISTER (CONTINUED)

- bit 2      Mode 0:  
**RB0DBEN:** Receive Buffer 0 Double-Buffer Enable bit  
1 = Receive Buffer 0 overflow will write to Receive Buffer 1  
0 = No Receive Buffer 0 overflow to Receive Buffer 1  
Mode 1, 2:  
**FILHIT<4:0>:** Filter Hit bit 2  
This bit combines with other bits to form filter acceptance bits<4:0>.
- bit 1      Mode 0:  
**JTOFF:** Jump Table Offset bit (read-only copy of RXB0DBEN)<sup>(2)</sup>  
1 = Allows jump table offset between 6 and 7  
0 = Allows jump table offset between 1 and 0  
Mode 1, 2:  
**FILHIT<4:0>:** Filter Hit bit 1  
This bit combines with other bits to form filter acceptance bits<4:0>.
- bit 0      Mode 0:  
**FILHIT0:** Filter Hit bit 0  
This bit indicates which acceptance filter enabled the message reception into Receive Buffer 0.  
1 = Acceptance Filter 1 (RXF1)  
0 = Acceptance Filter 0 (RXF0)  
Mode 1, 2:  
**FILHIT<4:0>:** Filter Hit bit 0  
This bit, in combination with FILHIT<4:1>, indicates which acceptance filter enabled the message reception into this receive buffer.  
01111 = Acceptance Filter 15 (RXF15)  
01110 = Acceptance Filter 14 (RXF14)  
...  
00000 = Acceptance Filter 0 (RXF0)
- Note 1:** This bit is set by the CAN module upon receiving a message and must be cleared by software after the buffer is read. As long as RXFUL is set, no new message will be loaded and the buffer will be considered full. After clearing the RXFUL flag, the PIR5 bit, RXB0IF, can be cleared. If RXB0IF is cleared, but RXFUL is not cleared, then RXB0IF is set again.
- 2:** This bit allows the same filter jump table for both RXB0CON and RXB1CON.

## REGISTER 34-14: RXB1CON: RECEIVE BUFFER 1 CONTROL REGISTER

| Mode 0 | R/C-0                | R/W-0 | R/W-0 | U-0 | R-0     | R/W-0   | R-0     | R-0     |
|--------|----------------------|-------|-------|-----|---------|---------|---------|---------|
|        | RXFUL <sup>(1)</sup> | RXM1  | RXM0  | —   | RXRTRRO | FILHIT2 | FILHIT1 | FILHIT0 |

| Mode 1,2 | R/C-0                | R/W-0 | R-0   | R-0     | R-0     | R-0     | R-0     | R-0     |
|----------|----------------------|-------|-------|---------|---------|---------|---------|---------|
|          | RXFUL <sup>(1)</sup> | RXM1  | RTRRO | FILHIT4 | FILHIT3 | FILHIT2 | FILHIT1 | FILHIT0 |
| bit 7    |                      |       |       |         |         |         |         | bit 0   |

|                   |                   |                                    |                    |
|-------------------|-------------------|------------------------------------|--------------------|
| <b>Legend:</b>    | C = Clearable bit |                                    |                    |
| R = Readable bit  | W = Writable bit  | U = Unimplemented bit, read as '0' |                    |
| -n = Value at POR | '1' = Bit is set  | '0' = Bit is cleared               | x = Bit is unknown |

- bit 7 **RXFUL:** Receive Full Status bit<sup>(1)</sup>  
 1 = Receive buffer contains a received message  
 0 = Receive buffer is open to receive a new message
- bit 6-5, 6 **Mode 0:**  
**RXM<1:0>:** Receive Buffer Mode bit 1 (combines with RXM0 to form RXM<1:0> bits, see bit 5)  
 11 = Receive all messages (including those with errors); filter criteria is ignored  
 10 = Receive only valid messages with extended identifier; EXIDEN in RXFnSIDL must be '1'  
 01 = Receive only valid messages with standard identifier, EXIDEN in RXFnSIDL must be '0'  
 00 = Receive all valid messages as per EXIDEN bit in RXFnSIDL register  
**Mode 1, 2:**  
**RXM1:** Receive Buffer Mode bit  
 1 = Receive all messages (including those with errors); acceptance filters are ignored  
 0 = Receive all valid messages as per acceptance filters
- bit 5 **Mode 0:**  
**RXM<1:0>:** Receive Buffer Mode bit 0 (combines with RXM1 to form RXM<1:0> bits, see bit 6)  
**Mode 1, 2:**  
**RTRRO:** Remote Transmission Request bit for Received Message (read-only)  
 1 = A remote transmission request is received  
 0 = A remote transmission request is not received
- bit 4 **Mode 0:**  
**FILHIT24:** Filter Hit bit 4  
**Mode 1, 2:**  
**FILHIT<4:0>:** Filter Hit bit 4  
 This bit combines with other bits to form the filter acceptance bits<4:0>.
- bit 3 **Mode 0:**  
**RXRTRRO:** Remote Transmission Request bit for Received Message (read-only)  
 1 = A remote transmission request is received  
 0 = A remote transmission request is not received  
**Mode 1, 2:**  
**FILHIT<4:0>:** Filter Hit bit 3  
 This bit combines with other bits to form the filter acceptance bits<4:0>.

**Note 1:** This bit is set by the CAN module upon receiving a message and must be cleared by software after the buffer is read. As long as RXFUL is set, no new message will be loaded and the buffer will be considered full.

## REGISTER 34-14: RXB1CON: RECEIVE BUFFER 1 CONTROL REGISTER (CONTINUED)

bit 2-0 Mode 0:

**FILHIT<2:0>:** Filter Hit bits

These bits indicate which acceptance filter enabled the last message reception into Receive Buffer 1.

111 = Reserved

110 = Reserved

101 = Acceptance Filter 5 (RXF5)

100 = Acceptance Filter 4 (RXF4)

011 = Acceptance Filter 3 (RXF3)

010 = Acceptance Filter 2 (RXF2)

001 = Acceptance Filter 1 (RXF1), only possible when RXB0DBEN bit is set

000 = Acceptance Filter 0 (RXF0), only possible when RXB0DBEN bit is set

Mode 1, 2:

**FILHIT<4:0>:** Filter Hit bits<2:0>

These bits, in combination with FILHIT<4:3>, indicate which acceptance filter enabled the message reception into this receive buffer.

01111 = Acceptance Filter 15 (RXF15)

01110 = Acceptance Filter 14 (RXF14)

...

00000 = Acceptance Filter 0 (RXF0)

**Note 1:** This bit is set by the CAN module upon receiving a message and must be cleared by software after the buffer is read. As long as RXFUL is set, no new message will be loaded and the buffer will be considered full.

## REGISTER 34-15: RXBnSIDH: RECEIVE BUFFER 'n' STANDARD IDENTIFIER REGISTERS, HIGH BYTE [0 ≤ n ≤ 1]

| R-x   | R-x  | R-x  | R-x  | R-x  | R-x  | R-x  | R-x   |
|-------|------|------|------|------|------|------|-------|
| SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3  |
| bit 7 |      |      |      |      |      |      | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-0

**SID<10:3>:** Standard Identifier bits (if EXID (RXBnSIDL<3>) = 0)

Extended Identifier bits, EID<28:21> (if EXID = 1).

## REGISTER 34-16: RXBnSIDL: RECEIVE BUFFER 'n' STANDARD IDENTIFIER REGISTERS, LOW BYTE [0 ≤ n ≤ 1]

| R-x   | R-x  | R-x  | R-x | R-x  | U-0 | R-x   | R-x   |
|-------|------|------|-----|------|-----|-------|-------|
| SID2  | SID1 | SID0 | SRR | EXID | —   | EID17 | EID16 |
| bit 7 |      |      |     |      |     |       | bit 0 |

### Legend:

R = Readable bit      W = Writable bit      U = Unimplemented bit, read as '0'  
-n = Value at POR      '1' = Bit is set      '0' = Bit is cleared      x = Bit is unknown

bit 7-5      **SID<2:0>**: Standard Identifier bits (if EXID = 0)  
Extended Identifier bits, EID<20:18> (if EXID = 1).  
bit 4      **SRR**: Substitute Remote Request bit  
bit 3      **EXID**: Extended Identifier bit  
1 = Received message is an extended data frame, SID<10:0> are EID<28:18>  
0 = Received message is a standard data frame  
bit 2      **Unimplemented**: Read as '0'  
bit 1-0      **EID<17:16>**: Extended Identifier bits

## REGISTER 34-17: RXBnEIDH: RECEIVE BUFFER 'n' EXTENDED IDENTIFIER REGISTERS, HIGH BYTE [0 ≤ n ≤ 1]

| R-x   | R-x   | R-x   | R-x   | R-x   | R-x   | R-x  | R-x   |
|-------|-------|-------|-------|-------|-------|------|-------|
| EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8  |
| bit 7 |       |       |       |       |       |      | bit 0 |

### Legend:

R = Readable bit      W = Writable bit      U = Unimplemented bit, read as '0'  
-n = Value at POR      '1' = Bit is set      '0' = Bit is cleared      x = Bit is unknown

bit 7-0      **EID<15:8>**: Extended Identifier bits

## REGISTER 34-18: RXBnEIDL: RECEIVE BUFFER 'n' EXTENDED IDENTIFIER REGISTERS, LOW BYTE [0 ≤ n ≤ 1]

| R-x   | R-x  | R-x  | R-x  | R-x  | R-x  | R-x  | R-x   |
|-------|------|------|------|------|------|------|-------|
| EID7  | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0  |
| bit 7 |      |      |      |      |      |      | bit 0 |

### Legend:

R = Readable bit      W = Writable bit      U = Unimplemented bit, read as '0'  
-n = Value at POR      '1' = Bit is set      '0' = Bit is cleared      x = Bit is unknown

bit 7-0      **EID<7:0>**: Extended Identifier bits

## REGISTER 34-19: RXBnDLC: RECEIVE BUFFER 'n' DATA LENGTH CODE REGISTERS [ $0 \leq n \leq 1$ ]

| U-0   | R-x   | R-x | R-x | R-x  | R-x  | R-x  | R-x   |
|-------|-------|-----|-----|------|------|------|-------|
| —     | RXRTR | RB1 | RB0 | DLC3 | DLC2 | DLC1 | DLC0  |
| bit 7 |       |     |     |      |      |      | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

- bit 7      **Unimplemented:** Read as '0'
- bit 6      **RXRTR:** Receiver Remote Transmission Request bit  
             1 = Remote transfer request  
             0 = No remote transfer request
- bit 5      **RB1:** Reserved bit 1  
             Reserved by CAN Spec and read as '0'.
- bit 4      **RB0:** Reserved bit 0  
             Reserved by CAN Spec and read as '0'.
- bit 3-0    **DLC<3:0>:** Data Length Code bits  
             1111 = Invalid  
             1110 = Invalid  
             1101 = Invalid  
             1100 = Invalid  
             1011 = Invalid  
             1010 = Invalid  
             1001 = Invalid  
             1000 = Data length = 8 bytes  
             0111 = Data length = 7 bytes  
             0110 = Data length = 6 bytes  
             0101 = Data length = 5 bytes  
             0100 = Data length = 4 bytes  
             0011 = Data length = 3 bytes  
             0010 = Data length = 2 bytes  
             0001 = Data length = 1 byte  
             0000 = Data length = 0 bytes

## REGISTER 34-20: RXBnDm: RECEIVE BUFFER 'n' DATA FIELD BYTE 'm' REGISTERS [ $0 \leq n \leq 1, 0 \leq m \leq 7$ ]

| R-x     | R-x     | R-x     | R-x     | R-x     | R-x     | R-x     | R-x     |
|---------|---------|---------|---------|---------|---------|---------|---------|
| RXBnDm7 | RXBnDm6 | RXBnDm5 | RXBnDm4 | RXBnDm3 | RXBnDm2 | RXBnDm1 | RXBnDm0 |
| bit 7   |         |         |         |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

- bit 7-0    **RXBnDm<7:0>:** Receive Buffer n Data Field Byte m bits (where  $0 \leq n < 1$  and  $0 < m < 7$ )  
             Each receive buffer has an array of registers. For example, Receive Buffer 0 has eight registers: RXB0D0 to RXB0D7.

## REGISTER 34-21: RXERRCNT: RECEIVE ERROR COUNT REGISTER

| R-0   | R-0  | R-0  | R-0  | R-0  | R-0  | R-0  | R-0   |
|-------|------|------|------|------|------|------|-------|
| REC7  | REC6 | REC5 | REC4 | REC3 | REC2 | REC1 | REC0  |
| bit 7 |      |      |      |      |      |      | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-0

**REC<7:0>:** Receive Error Counter bits

This register contains the receive error value as defined by the CAN specifications. When RXERRCNT > 127, the module will go into an error-passive state. RXERRCNT does not have the ability to put the module in "bus-off" state.

## EXAMPLE 34-5: READING A CAN MESSAGE

```

; Need to read a pending message from RXB0 buffer.
; To receive any message, filter, mask and RXM1:RXM0 bits in RXB0CON registers must be
; programmed correctly.
;
; Make sure that there is a message pending in RXB0.
BTFS RXB0CON, RXFUL          ; Does RXB0 contain a message?
BRA   NoMessage              ; No. Handle this situation...
; We have verified that a message is pending in RXB0 buffer.
; If this buffer can receive both Standard or Extended Identifier messages,
; identify type of message received.
BTFS RXB0SIDL, EXID          ; Is this Extended Identifier?
BRA   StandardMessage        ; No. This is Standard Identifier message.
                                ; Yes. This is Extended Identifier message.
; Read all 29-bits of Extended Identifier message.
...
; Now read all data bytes
MOVFF RXB0DO, MY_DATA_BYTE1
...
; Once entire message is read, mark the RXB0 that it is read and no longer FULL.
BCF   RXB0CON, RXFUL          ; This will allow CAN Module to load new messages
                                ; into this buffer.
...

```

## 34.15.3.1 Programmable TX/RX and Auto-RTR Buffers

The ECAN module contains six message buffers that can be programmed as transmit or receive buffers. Any of these buffers can also be programmed to automatically handle RTR messages.

**Note:** These registers are not used in Mode 0.

### REGISTER 34-22: BnCON: TX/RX BUFFER 'n' CONTROL REGISTERS IN RECEIVE MODE [0 ≤ n ≤ 5, TXnEN (BSEL0<n>) = 0]<sup>(1)</sup>

| R/W-0                | R/W-0 | R-0     | R-0     | R-0     | R-0     | R-0     | R-0     |
|----------------------|-------|---------|---------|---------|---------|---------|---------|
| RXFUL <sup>(2)</sup> | RXM1  | RXRTRRO | FILHIT4 | FILHIT3 | FILHIT2 | FILHIT1 | FILHIT0 |
| bit 7                |       |         |         |         |         |         | bit 0   |

#### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7 **RXFUL:** Receive Full Status bit<sup>(2)</sup>

1 = Receive buffer contains a received message

0 = Receive buffer is open to receive a new message

bit 6 **RXM1:** Receive Buffer Mode bit

1 = Receive all messages including partial and invalid (acceptance filters are ignored)

0 = Receive all valid messages as per acceptance filters

bit 5 **RXRTRRO:** Read-Only Remote Transmission Request for Received Message bit

1 = Received message is a remote transmission request

0 = Received message is not a remote transmission request

bit 4-0 **FILHIT<4:0>:** Filter Hit bits

These bits indicate which acceptance filter enabled the last message reception into this buffer.

01111 = Acceptance Filter 15 (RXF15)

01110 = Acceptance Filter 14 (RXF14)

...

00001 = Acceptance Filter 1 (RXF1)

00000 = Acceptance Filter 0 (RXF0)

**Note 1:** These registers are available in Mode 1 and 2 only.

**2:** This bit is set by the CAN module upon receiving a message and must be cleared by software after the buffer is read. As long as RXFUL is set, no new message will be loaded and the buffer will be considered full.

## REGISTER 34-23: BnCON: TX/RX BUFFER 'n' CONTROL REGISTERS IN TRANSMIT MODE [0 ≤ n ≤ 5, TXnEN (BSEL0<n>) = 1]<sup>(1)</sup>

| R/W-0                | R-0                  | R-0                   | R-0                  | R/W-0                  | R/W-0 | R/W-0                 | R/W-0                 |
|----------------------|----------------------|-----------------------|----------------------|------------------------|-------|-----------------------|-----------------------|
| TXBIF <sup>(3)</sup> | TXABT <sup>(3)</sup> | TXLARB <sup>(3)</sup> | TXERR <sup>(3)</sup> | TXREQ <sup>(2,4)</sup> | RTREN | TXPRI1 <sup>(5)</sup> | TXPRI0 <sup>(5)</sup> |
| bit 7                |                      |                       |                      |                        |       |                       | bit 0                 |

### Legend:

R = Readable bit                      W = Writable bit                      U = Unimplemented bit, read as '0'  
-n = Value at POR                      '1' = Bit is set                      '0' = Bit is cleared                      x = Bit is unknown

- bit 7                      **TXBIF:** Transmit Buffer Interrupt Flag bit<sup>(3)</sup>  
1 = A message was successfully transmitted  
0 = No message was transmitted
- bit 6                      **TXABT:** Transmission Aborted Status bit<sup>(3)</sup>  
1 = Message was aborted  
0 = Message was not aborted
- bit 5                      **TXLARB:** Transmission Lost Arbitration Status bit<sup>(3)</sup>  
1 = Message lost arbitration while being sent  
0 = Message did not lose arbitration while being sent
- bit 4                      **TXERR:** Transmission Error Detected Status bit<sup>(3)</sup>  
1 = A bus error occurred while the message was being sent  
0 = A bus error did not occur while the message was being sent
- bit 3                      **TXREQ:** Transmit Request Status bit<sup>(2,4)</sup>  
1 = Requests sending a message; clears the TXABT, TXLARB and TXERR bits  
0 = Automatically cleared when the message is successfully sent
- bit 2                      **RTREN:** Automatic Remote Transmission Request Enable bit  
1 = When a remote transmission request is received, TXREQ will be automatically set  
0 = When a remote transmission request is received, TXREQ will be unaffected
- bit 1-0                      **TXPRI<1:0>:** Transmit Priority bits<sup>(5)</sup>  
11 = Priority Level 3 (highest priority)  
10 = Priority Level 2  
01 = Priority Level 1  
00 = Priority Level 0 (lowest priority)

- Note 1:** These registers are available in Mode 1 and 2 only.
- 2:** Clearing this bit in software while the bit is set will request a message abort.
- 3:** This bit is automatically cleared when TXREQ is set.
- 4:** While TXREQ is set or a transmission is in progress, Transmit Buffer registers remain read-only.
- 5:** These bits set the order in which the Transmit Buffer register will be transferred. They do not alter the CAN message identifier.



**REGISTER 34-24: BnSIDH: TX/RX BUFFER 'n' STANDARD IDENTIFIER REGISTERS,  
HIGH BYTE IN RECEIVE MODE [ $0 \leq n \leq 5$ , TXnEN (BSEL0<n>) = 0]<sup>(1)</sup>**

| R-x   | R-x  | R-x  | R-x  | R-x  | R-x  | R-x  | R-x   |
|-------|------|------|------|------|------|------|-------|
| SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3  |
| bit 7 |      |      |      |      |      |      | bit 0 |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-0 **SID<10:3>**: Standard Identifier bits (if EXIDE (BnSIDL<3>) = 0)  
Extended Identifier bits, EID<28:21> (if EXIDE = 1).

**Note 1:** These registers are available in Mode 1 and 2 only.

**REGISTER 34-25: BnSIDH: TX/RX BUFFER 'n' STANDARD IDENTIFIER REGISTERS,  
HIGH BYTE IN TRANSMIT MODE [ $0 \leq n \leq 5$ , TXnEN (BSEL0<n>) = 1]<sup>(1)</sup>**

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|-------|-------|-------|-------|-------|-------|-------|-------|
| SID10 | SID9  | SID8  | SID7  | SID6  | SID5  | SID4  | SID3  |
| bit 7 |       |       |       |       |       |       | bit 0 |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-0 **SID<10:3>**: Standard Identifier bits (if EXIDE (BnSIDL<3>) = 0)  
Extended Identifier bits, EID<28:21> (if EXIDE = 1).

**Note 1:** These registers are available in Mode 1 and 2 only.

## REGISTER 34-26: BnSIDL: TX/RX BUFFER 'n' STANDARD IDENTIFIER REGISTERS, LOW BYTE IN RECEIVE MODE [ $0 \leq n \leq 5$ , TXnEN (BSEL0<n>) = 0]<sup>(1)</sup>

|       |      |      |     |       |     |       |       |
|-------|------|------|-----|-------|-----|-------|-------|
| R-x   | R-x  | R-x  | R-x | R-x   | U-0 | R-x   | R-x   |
| SID2  | SID1 | SID0 | SRR | EXIDE | —   | EID17 | EID16 |
| bit 7 |      |      |     |       |     |       | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-5 **SID<2:0>**: Standard Identifier bits (if EXID = 0)

Extended Identifier bits, EID<20:18> (if EXID = 1).

bit 4 **SRR**: Substitute Remote Transmission Request bit

This bit is always '1' when EXID = 1 or equal to the value of RXRTRRO (BnCON<5>) when EXID = 0.

bit 3 **EXIDE**: Extended Identifier Enable bit

1 = Received message is an extended identifier frame (SID<10:0> are EID<28:18>)

0 = Received message is a standard identifier frame

bit 2 **Unimplemented**: Read as '0'

bit 1-0 **EID<17:16>**: Extended Identifier bits

**Note 1:** These registers are available in Mode 1 and 2 only.

## REGISTER 34-27: BnSIDL: TX/RX BUFFER 'n' STANDARD IDENTIFIER REGISTERS, LOW BYTE IN TRANSMIT MODE [ $0 \leq n \leq 5$ , TXnEN (BSEL0<n>) = 1]<sup>(1)</sup>

|       |       |       |     |       |     |       |       |
|-------|-------|-------|-----|-------|-----|-------|-------|
| R/W-x | R/W-x | R/W-x | U-0 | R/W-x | U-0 | R/W-x | R/W-x |
| SID2  | SID1  | SID0  | —   | EXIDE | —   | EID17 | EID16 |
| bit 7 |       |       |     |       |     |       | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-5 **SID<2:0>**: Standard Identifier bits (if EXIDE (TXBnSIDL<3>) = 0)

Extended Identifier bits, EID<20:18> (if EXIDE = 1).

bit 4 **Unimplemented**: Read as '0'

bit 3 **EXIDE**: Extended Identifier Enable bit

1 = Message will transmit extended ID, SID<10:0> bits become EID<28:18>

0 = Received will transmit standard ID, EID<17:0> are ignored

bit 2 **Unimplemented**: Read as '0'

bit 1-0 **EID<17:16>**: Extended Identifier bits

**Note 1:** These registers are available in Mode 1 and 2 only.

**REGISTER 34-28: BnEIDH: TX/RX BUFFER 'n' EXTENDED IDENTIFIER REGISTERS,  
HIGH BYTE IN RECEIVE MODE [ $0 \leq n \leq 5$ , TXnEN (BSEL0<n>) = 0]<sup>(1)</sup>**

|       |       |       |       |       |       |      |       |
|-------|-------|-------|-------|-------|-------|------|-------|
| R-x   | R-x   | R-x   | R-x   | R-x   | R-x   | R-x  | R-x   |
| EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8  |
| bit 7 |       |       |       |       |       |      | bit 0 |

**Legend:**

R = Readable bit                      W = Writable bit                      U = Unimplemented bit, read as '0'  
-n = Value at POR                      '1' = Bit is set                      '0' = Bit is cleared                      x = Bit is unknown

bit 7-0                      **EID<15:8>**: Extended Identifier bits

**Note 1:** These registers are available in Mode 1 and 2 only.

**REGISTER 34-29: BnEIDH: TX/RX BUFFER 'n' EXTENDED IDENTIFIER REGISTERS,  
HIGH BYTE IN TRANSMIT MODE [ $0 \leq n \leq 5$ , TXnEN (BSEL0<n>) = 1]<sup>(1)</sup>**

|       |       |       |       |       |       |       |       |
|-------|-------|-------|-------|-------|-------|-------|-------|
| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
| EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9  | EID8  |
| bit 7 |       |       |       |       |       |       | bit 0 |

**Legend:**

R = Readable bit                      W = Writable bit                      U = Unimplemented bit, read as '0'  
-n = Value at POR                      '1' = Bit is set                      '0' = Bit is cleared                      x = Bit is unknown

bit 7-0                      **EID<15:8>**: Extended Identifier bits

**Note 1:** These registers are available in Mode 1 and 2 only.

**REGISTER 34-30: BnEIDL: TX/RX BUFFER 'n' EXTENDED IDENTIFIER REGISTERS,  
LOW BYTE IN RECEIVE MODE [ $0 \leq n \leq 5$ , TXnEN (BSEL<n>) = 0]<sup>(1)</sup>**

|       |      |      |      |      |      |      |       |
|-------|------|------|------|------|------|------|-------|
| R-x   | R-x  | R-x  | R-x  | R-x  | R-x  | R-x  | R-x   |
| EID7  | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0  |
| bit 7 |      |      |      |      |      |      | bit 0 |

**Legend:**

R = Readable bit                      W = Writable bit                      U = Unimplemented bit, read as '0'  
-n = Value at POR                      '1' = Bit is set                      '0' = Bit is cleared                      x = Bit is unknown

bit 7-0                      **EID<7:0>**: Extended Identifier bits

**Note 1:** These registers are available in Mode 1 and 2 only.

## REGISTER 34-31: BnEIDL: TX/RX BUFFER 'n' EXTENDED IDENTIFIER REGISTERS, LOW BYTE IN RECEIVE MODE [ $0 \leq n \leq 5$ , TXnEN (BSEL<n>) = 1]<sup>(1)</sup>

|       |       |       |       |       |       |       |       |
|-------|-------|-------|-------|-------|-------|-------|-------|
| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
| EID7  | EID6  | EID5  | FEID4 | EID3  | EID2  | EID1  | EID0  |
| bit 7 |       |       |       |       |       |       | bit 0 |

### Legend:

R = Readable bit      W = Writable bit      U = Unimplemented bit, read as '0'  
 -n = Value at POR      '1' = Bit is set      '0' = Bit is cleared      x = Bit is unknown

bit 7-0      **EID<7:0>**: Extended Identifier bits

**Note 1:** These registers are available in Mode 1 and 2 only.

## REGISTER 34-32: BnDm: TX/RX BUFFER 'n' DATA FIELD BYTE 'm' REGISTERS IN RECEIVE MODE [ $0 \leq n \leq 5$ , $0 \leq m \leq 7$ , TXnEN (BSEL<n>) = 0]<sup>(1)</sup>

|       |       |       |       |       |       |       |       |
|-------|-------|-------|-------|-------|-------|-------|-------|
| R-x   | R-x   | R-x   | R-x   | R-x   | R-x   | R-x   | R-x   |
| BnDm7 | BnDm6 | BnDm5 | BnDm4 | BnDm3 | BnDm2 | BnDm1 | BnDm0 |
| bit 7 |       |       |       |       |       |       | bit 0 |

### Legend:

R = Readable bit      W = Writable bit      U = Unimplemented bit, read as '0'  
 -n = Value at POR      '1' = Bit is set      '0' = Bit is cleared      x = Bit is unknown

bit 7-0      **BnDm<7:0>**: Receive Buffer n Data Field Byte m bits (where  $0 \leq n < 3$  and  $0 < m < 8$ )  
 Each receive buffer has an array of registers. For example, Receive Buffer 0 has 7 registers: B0D0 to B0D7.

**Note 1:** These registers are available in Mode 1 and 2 only.

## REGISTER 34-33: BnDm: TX/RX BUFFER 'n' DATA FIELD BYTE 'm' REGISTERS IN TRANSMIT MODE [ $0 \leq n \leq 5$ , $0 \leq m \leq 7$ , TXnEN (BSEL<n>) = 1]<sup>(1)</sup>

|       |       |       |       |       |       |       |       |
|-------|-------|-------|-------|-------|-------|-------|-------|
| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
| BnDm7 | BnDm6 | BnDm5 | BnDm4 | BnDm3 | BnDm2 | BnDm1 | BnDm0 |
| bit 7 |       |       |       |       |       |       | bit 0 |

### Legend:

R = Readable bit      W = Writable bit      U = Unimplemented bit, read as '0'  
 -n = Value at POR      '1' = Bit is set      '0' = Bit is cleared      x = Bit is unknown

bit 7-0      **BnDm<7:0>**: Transmit Buffer n Data Field Byte m bits (where  $0 \leq n < 3$  and  $0 < m < 8$ )  
 Each transmit buffer has an array of registers. For example, Transmit Buffer 0 has 7 registers: TXB0D0 to TXB0D7.

**Note 1:** These registers are available in Mode 1 and 2 only.

## REGISTER 34-34: BnDLC: TX/RX BUFFER 'n' DATA LENGTH CODE REGISTERS IN RECEIVE MODE [0 ≤ n ≤ 5, TXnEN (BSEL<n>) = 0]<sup>(1)</sup>

| U-0   | R-x   | R-x | R-x | R-x  | R-x  | R-x  | R-x   |
|-------|-------|-----|-----|------|------|------|-------|
| —     | RXRTR | RB1 | RB0 | DLC3 | DLC2 | DLC1 | DLC0  |
| bit 7 |       |     |     |      |      |      | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7 **Unimplemented:** Read as '0'

bit 6 **RXRTR:** Receiver Remote Transmission Request bit

1 = This is a remote transmission request

0 = This is not a remote transmission request

bit 5 **RB1:** Reserved bit 1

Reserved by CAN Spec and read as '0'.

bit 4 **RB0:** Reserved bit 0

Reserved by CAN Spec and read as '0'.

bit 3-0 **DLC<3:0>:** Data Length Code bits

1111 = Reserved

1110 = Reserved

1101 = Reserved

1100 = Reserved

1011 = Reserved

1010 = Reserved

1001 = Reserved

1000 = Data length = 8 bytes

0111 = Data length = 7 bytes

0110 = Data length = 6 bytes

0101 = Data length = 5 bytes

0100 = Data length = 4 bytes

0011 = Data length = 3 bytes

0010 = Data length = 2 bytes

0001 = Data length = 1 byte

0000 = Data length = 0 bytes

**Note 1:** These registers are available in Mode 1 and 2 only.

## REGISTER 34-35: BnDLC: TX/RX BUFFER 'n' DATA LENGTH CODE REGISTERS IN TRANSMIT MODE [0 ≤ n ≤ 5, TXnEN (BSEL<n>) = 1]<sup>(1)</sup>

| U-0   | R/W-x | U-0 | U-0 | R/W-x | R/W-x | R/W-x | R/W-x |
|-------|-------|-----|-----|-------|-------|-------|-------|
| —     | TXRTR | —   | —   | DLC3  | DLC2  | DLC1  | DLC0  |
| bit 7 |       |     |     | bit 0 |       |       |       |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7 **Unimplemented:** Read as '0'

bit 6 **TXRTR:** Transmitter Remote Transmission Request bit

1 = Transmitted message will have the RTR bit set

0 = Transmitted message will have the RTR bit cleared

bit 5-4 **Unimplemented:** Read as '0'

bit 3-0 **DLC<3:0>:** Data Length Code bits

1111-1001 = Reserved

1000 = Data length = 8 bytes

0111 = Data length = 7 bytes

0110 = Data length = 6 bytes

0101 = Data length = 5 bytes

0100 = Data length = 4 bytes

0011 = Data length = 3 bytes

0010 = Data length = 2 bytes

0001 = Data length = 1 byte

0000 = Data length = 0 bytes

**Note 1:** These registers are available in Mode 1 and 2 only.

## REGISTER 34-36: BSEL0: BUFFER SELECT REGISTER 0<sup>(1)</sup>

| R/W-0  | R/W-0  | R/W-0  | R/W-0  | R/W-0  | R/W-0  | U-0   | U-0 |
|--------|--------|--------|--------|--------|--------|-------|-----|
| B5TXEN | B4TXEN | B3TXEN | B2TXEN | B1TXEN | B0TXEN | —     | —   |
| bit 7  |        |        |        |        |        | bit 0 |     |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-2 **B<5:0>TXEN:** Buffer 5 to Buffer 0 Transmit Enable bits

1 = Buffer is configured in Transmit mode

0 = Buffer is configured in Receive mode

bit 1-0 **Unimplemented:** Read as '0'

**Note 1:** These registers are available in Mode 1 and 2 only.

## 34.15.3.2 Message Acceptance Filters and Masks

This section describes the message acceptance filters and masks for the CAN receive buffers.

### REGISTER 34-37: RXF<sub>n</sub>SIDH: RECEIVE ACCEPTANCE FILTER 'n' STANDARD IDENTIFIER FILTER REGISTERS, HIGH BYTE [0 ≤ n ≤ 15]<sup>(1)</sup>

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|-------|-------|-------|-------|-------|-------|-------|-------|
| SID10 | SID9  | SID8  | SID7  | SID6  | SID5  | SID4  | SID3  |
| bit 7 |       |       |       |       |       |       | bit 0 |

#### Legend:

R = Readable bit                      W = Writable bit                      U = Unimplemented bit, read as '0'  
 -n = Value at POR                      '1' = Bit is set                      '0' = Bit is cleared                      x = Bit is unknown

bit 7-0                      **SID<10:3>**: Standard Identifier Filter bits (if EXIDEN = 0)  
 Extended Identifier Filter bits, EID<28:21> (if EXIDEN = 1).

**Note 1:** Registers, RXF6SIDH:RXF15SIDH, are available in Mode 1 and 2 only.

### REGISTER 34-38: RXF<sub>n</sub>SIDL: RECEIVE ACCEPTANCE FILTER 'n' STANDARD IDENTIFIER FILTER REGISTERS, LOW BYTE [0 ≤ n ≤ 15]<sup>(1)</sup>

| R/W-x | R/W-x | R/W-x | U-0 | R/W-x                 | U-0 | R/W-x | R/W-x |
|-------|-------|-------|-----|-----------------------|-----|-------|-------|
| SID2  | SID1  | SID0  | —   | EXIDEN <sup>(2)</sup> | —   | EID17 | EID16 |
| bit 7 |       |       |     |                       |     |       | bit 0 |

#### Legend:

R = Readable bit                      W = Writable bit                      U = Unimplemented bit, read as '0'  
 -n = Value at POR                      '1' = Bit is set                      '0' = Bit is cleared                      x = Bit is unknown

bit 7-5                      **SID<2:0>**: Standard Identifier Filter bits (if EXIDEN = 0)  
 Extended Identifier Filter bits, EID<20:18> (if EXIDEN = 1).

bit 4                      **Unimplemented**: Read as '0'

bit 3                      **EXIDEN**: Extended Identifier Filter Enable bit<sup>(2)</sup>  
 1 = Filter will only accept extended ID messages  
 0 = Filter will only accept standard ID messages

bit 2                      **Unimplemented**: Read as '0'

bit 1-0                      **EID<17:16>**: Extended Identifier Filter bits

**Note 1:** Registers, RXF6SIDL:RXF15SIDL, are available in Mode 1 and 2 only.

**2:** In Mode 0, this bit must be set/cleared as required, irrespective of corresponding mask register value.

## REGISTER 34-39: RXFnEIDH: RECEIVE ACCEPTANCE FILTER 'n' EXTENDED IDENTIFIER REGISTERS, HIGH BYTE [0 ≤ n ≤ 15]<sup>(1)</sup>

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|-------|-------|-------|-------|-------|-------|-------|-------|
| EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9  | EID8  |
| bit 7 |       |       |       |       |       |       | bit 0 |

### Legend:

R = Readable bit      W = Writable bit      U = Unimplemented bit, read as '0'  
 -n = Value at POR      '1' = Bit is set      '0' = Bit is cleared      x = Bit is unknown

bit 7-0      **EID<15:8>**: Extended Identifier Filter bits

**Note 1:** Registers, RXF6EIDH:RXF15EIDH, are available in Mode 1 and 2 only.

## REGISTER 34-40: RXFnEIDL: RECEIVE ACCEPTANCE FILTER 'n' EXTENDED IDENTIFIER REGISTERS, LOW BYTE [0 ≤ n ≤ 15]<sup>(1)</sup>

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|-------|-------|-------|-------|-------|-------|-------|-------|
| EID7  | EID6  | EID5  | EID4  | EID3  | EID2  | EID1  | EID0  |
| bit 7 |       |       |       |       |       |       | bit 0 |

### Legend:

R = Readable bit      W = Writable bit      U = Unimplemented bit, read as '0'  
 -n = Value at POR      '1' = Bit is set      '0' = Bit is cleared      x = Bit is unknown

bit 7-0      **EID<7:0>**: Extended Identifier Filter bits

**Note 1:** Registers, RXF6EIDL:RXF15EIDL, are available in Mode 1 and 2 only.

## REGISTER 34-41: RXMnSIDH: RECEIVE ACCEPTANCE MASK 'n' STANDARD IDENTIFIER MASK REGISTERS, HIGH BYTE [0 ≤ n ≤ 1]

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|-------|-------|-------|-------|-------|-------|-------|-------|
| SID10 | SID9  | SID8  | SID7  | SID6  | SID5  | SID4  | SID3  |
| bit 7 |       |       |       |       |       |       | bit 0 |

### Legend:

R = Readable bit      W = Writable bit      U = Unimplemented bit, read as '0'  
 -n = Value at POR      '1' = Bit is set      '0' = Bit is cleared      x = Bit is unknown

bit 7-0      **SID<10:3>**: Standard Identifier Mask bits or Extended Identifier Mask bits (EID<28:21>)



## REGISTER 34-42: RXMnSIDL: RECEIVE ACCEPTANCE MASK 'n' STANDARD IDENTIFIER MASK REGISTERS, LOW BYTE [ $0 \leq n \leq 1$ ]

|       |       |       |     |                       |     |       |       |
|-------|-------|-------|-----|-----------------------|-----|-------|-------|
| R/W-x | R/W-x | R/W-x | U-0 | R/W-0                 | U-0 | R/W-x | R/W-x |
| SID2  | SID1  | SID0  | —   | EXIDEN <sup>(1)</sup> | —   | EID17 | EID16 |
| bit 7 |       |       |     |                       |     |       | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-5 **SID<2:0>**: Standard Identifier Mask bits or Extended Identifier Mask bits (EID<20:18>)

bit 4 **Unimplemented**: Read as '0'

bit 3 **Mode 0:**  
**Unimplemented**: Read as '0'

**Mode 1, 2:**

**EXIDEN**: Extended Identifier Filter Enable Mask bit<sup>(1)</sup>

1 = Messages selected by the EXIDEN bit in RXFnSIDL will be accepted

0 = Both standard and extended identifier messages will be accepted

bit 2 **Unimplemented**: Read as '0'

bit 1-0 **EID<17:16>**: Extended Identifier Mask bits

**Note 1:** This bit is available in Mode 1 and 2 only.

## REGISTER 34-43: RXMnEIDH: RECEIVE ACCEPTANCE MASK 'n' EXTENDED IDENTIFIER MASK REGISTERS, HIGH BYTE [ $0 \leq n \leq 1$ ]

|       |       |       |       |       |       |       |       |
|-------|-------|-------|-------|-------|-------|-------|-------|
| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
| EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9  | EID8  |
| bit 7 |       |       |       |       |       |       | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-0 **EID<15:8>**: Extended Identifier Mask bits



## REGISTER 34-46: SDFLC: STANDARD DATA BYTES FILTER LENGTH COUNT REGISTER<sup>(1)</sup>

|       |     |     |       |       |       |       |       |
|-------|-----|-----|-------|-------|-------|-------|-------|
| U-0   | U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| —     | —   | —   | FLC4  | FLC3  | FLC2  | FLC1  | FLC0  |
| bit 7 |     |     |       |       |       |       | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-5 **Unimplemented:** Read as '0'

bit 4-0 **FLC<4:0>:** Filter Length Count bits

#### Mode 0:

Not used; forced to '00000'.

#### Mode 1, 2:

00000-10010 = 0

18 bits are available for standard data byte filter. Actual number of bits used depends on the DLC<3:0> bits (RXBnDLC<3:0> or BnDLC<3:0> if configured as RX buffer) of the message being received.

If DLC<3:0> = 0000

No bits will be compared with incoming data bits.

If DLC<3:0> = 0001

Up to 8 data bits of RXFnEID<7:0>, as determined by FLC<2:0>, will be compared with the corresponding number of data bits of the incoming message.

If DLC<3:0> = 0010

Up to 16 data bits of RXFnEID<15:0>, as determined by FLC<3:0>, will be compared with the corresponding number of data bits of the incoming message.

If DLC<3:0> = 0011

Up to 18 data bits of RXFnEID<17:0>, as determined by FLC<4:0>, will be compared with the corresponding number of data bits of the incoming message.

**Note 1:** This register is available in Mode 1 and 2 only.

## REGISTER 34-47: RXFBCONn: RECEIVE FILTER BUFFER CONTROL REGISTER 'n' <sup>(1)</sup>

|          |         |         |         |         |         |         |         |         |
|----------|---------|---------|---------|---------|---------|---------|---------|---------|
| RXFBCON0 | R/W-0   | R/W-0   | R/W-0   | R/W-0   | R/W-0   | R/W-0   | R/W-0   | R/W-0   |
|          | F1BP_3  | F1BP_2  | F1BP_1  | F1BP_0  | F0BP_3  | F0BP_2  | F0BP_1  | F0BP_0  |
| RXFBCON1 | R/W-0   | R/W-0   | R/W-0   | R/W-1   | R/W-0   | R/W-0   | R/W-0   | R/W-1   |
|          | F3BP_3  | F3BP_2  | F3BP_1  | F3BP_0  | F2BP_3  | F2BP_2  | F2BP_1  | F2BP_0  |
| RXFBCON2 | R/W-0   | R/W-0   | R/W-0   | R/W-1   | R/W-0   | R/W-0   | R/W-0   | R/W-1   |
|          | F5BP_3  | F5BP_2  | F5BP_1  | F5BP_0  | F4BP_3  | F4BP_2  | F4BP_1  | F4BP_0  |
| RXFBCON3 | R/W-0   | R/W-0   | R/W-0   | R/W-0   | R/W-0   | R/W-0   | R/W-0   | R/W-0   |
|          | F7BP_3  | F7BP_2  | F7BP_1  | F7BP_0  | F6BP_3  | F6BP_2  | F6BP_1  | F6BP_0  |
| RXFBCON4 | R/W-0   | R/W-0   | R/W-0   | R/W-0   | R/W-0   | R/W-0   | R/W-0   | R/W-0   |
|          | F9BP_3  | F9BP_2  | F9BP_1  | F9BP_0  | F8BP_3  | F8BP_2  | F8BP_1  | F8BP_0  |
| RXFBCON5 | R/W-0   | R/W-0   | R/W-0   | R/W-0   | R/W-0   | R/W-0   | R/W-0   | R/W-0   |
|          | F11BP_3 | F11BP_2 | F11BP_1 | F11BP_0 | F10BP_3 | F10BP_2 | F10BP_1 | F10BP_0 |
| RXFBCON6 | R/W-0   | R/W-0   | R/W-0   | R/W-0   | R/W-0   | R/W-0   | R/W-0   | R/W-0   |
|          | F13BP_3 | F13BP_2 | F13BP_1 | F13BP_0 | F12BP_3 | F12BP_2 | F12BP_1 | F12BP_0 |
| RXFBCON7 | R/W-0   | R/W-0   | R/W-0   | R/W-0   | R/W-0   | R/W-0   | R/W-0   | R/W-0   |
|          | F15BP_3 | F15BP_2 | F15BP_1 | F15BP_0 | F14BP_3 | F14BP_2 | F14BP_1 | F14BP_0 |
|          | bit 7   |         |         |         |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-0 **F<15:2>BP\_<3:0>**: Filter n Buffer Pointer Nibble bits

0000 = Filter n is associated with RXB0

0001 = Filter n is associated with RXB1

0010 = Filter n is associated with B0

0011 = Filter n is associated with B1

...

0111 = Filter n is associated with B5

1111-1000 = Reserved

**Note 1:** This register is available in Mode 1 and 2 only.

## REGISTER 34-48: MSEL0: MASK SELECT REGISTER 0<sup>(1)</sup>

|        |        |        |        |        |        |        |        |
|--------|--------|--------|--------|--------|--------|--------|--------|
| R/W-0  | R/W-1  | R/W-0  | R/W-1  | R/W-0  | R/W-0  | R/W-0  | R/W-0  |
| FIL3_1 | FIL3_0 | FIL2_1 | FIL2_0 | FIL1_1 | FIL1_0 | FIL0_1 | FIL0_0 |
| bit 7  |        |        |        |        |        |        | bit 0  |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-6 **FIL3\_<1:0>**: Filter 3 Select bits 1 and 0

11 = No mask

10 = Filter 15

01 = Acceptance Mask 1

00 = Acceptance Mask 0

bit 5-4 **FIL2\_<1:0>**: Filter 2 Select bits 1 and 0

11 = No mask

10 = Filter 15

01 = Acceptance Mask 1

00 = Acceptance Mask 0

bit 3-2 **FIL1\_<1:0>**: Filter 1 Select bits 1 and 0

11 = No mask

10 = Filter 15

01 = Acceptance Mask 1

00 = Acceptance Mask 0

bit 1-0 **FIL0\_<1:0>**: Filter 0 Select bits 1 and 0

11 = No mask

10 = Filter 15

01 = Acceptance Mask 1

00 = Acceptance Mask 0

**Note 1:** This register is available in Mode 1 and 2 only.

## REGISTER 34-49: MSEL1: MASK SELECT REGISTER 1<sup>(1)</sup>

|        |        |        |        |        |        |        |        |
|--------|--------|--------|--------|--------|--------|--------|--------|
| R/W-0  | R/W-0  | R/W-0  | R/W-0  | R/W-0  | R/W-1  | R/W-0  | R/W-1  |
| FIL7_1 | FIL7_0 | FIL6_1 | FIL6_0 | FIL5_1 | FIL5_0 | FIL4_1 | FIL4_0 |
| bit 7  |        |        |        |        |        |        | bit 0  |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-6 **FIL7\_<1:0>**: Filter 7 Select bits 1 and 0

11 = No mask

10 = Filter 15

01 = Acceptance Mask 1

00 = Acceptance Mask 0

bit 5-4 **FIL6\_<1:0>**: Filter 6 Select bits 1 and 0

11 = No mask

10 = Filter 15

01 = Acceptance Mask 1

00 = Acceptance Mask 0

bit 3-2 **FIL5\_<1:0>**: Filter 5 Select bits 1 and 0

11 = No mask

10 = Filter 15

01 = Acceptance Mask 1

00 = Acceptance Mask 0

bit 1-0 **FIL4\_<1:0>**: Filter 4 Select bits 1 and 0

11 = No mask

10 = Filter 15

01 = Acceptance Mask 1

00 = Acceptance Mask 0

**Note 1:** This register is available in Mode 1 and 2 only.

## REGISTER 34-50: MSEL2: MASK SELECT REGISTER 2<sup>(1)</sup>

|         |         |         |         |        |        |        |        |
|---------|---------|---------|---------|--------|--------|--------|--------|
| R/W-0   | R/W-0   | R/W-0   | R/W-0   | R/W-0  | R/W-0  | R/W-0  | R/W-0  |
| FIL11_1 | FIL11_0 | FIL10_1 | FIL10_0 | FIL9_1 | FIL9_0 | FIL8_1 | FIL8_0 |
| bit 7   |         |         |         |        |        |        | bit 0  |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-6 **FIL11\_<1:0>**: Filter 11 Select bits 1 and 0

11 = No mask

10 = Filter 15

01 = Acceptance Mask 1

00 = Acceptance Mask 0

bit 5-4 **FIL10\_<1:0>**: Filter 10 Select bits 1 and 0

11 = No mask

10 = Filter 15

01 = Acceptance Mask 1

00 = Acceptance Mask 0

bit 3-2 **FIL9\_<1:0>**: Filter 9 Select bits 1 and 0

11 = No mask

10 = Filter 15

01 = Acceptance Mask 1

00 = Acceptance Mask 0

bit 1-0 **FIL8\_<1:0>**: Filter 8 Select bits 1 and 0

11 = No mask

10 = Filter 15

01 = Acceptance Mask 1

00 = Acceptance Mask 0

**Note 1:** This register is available in Mode 1 and 2 only.

## REGISTER 34-51: MSEL3: MASK SELECT REGISTER 3<sup>(1)</sup>

|         |         |         |         |         |         |         |         |
|---------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0   | R/W-0   | R/W-0   | R/W-0   | R/W-0   | R/W-0   | R/W-0   | R/W-0   |
| FIL15_1 | FIL15_0 | FIL14_1 | FIL14_0 | FIL13_1 | FIL13_0 | FIL12_1 | FIL12_0 |
| bit 7   |         |         |         |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-6 **FIL15\_<1:0>**: Filter 15 Select bits 1 and 0

11 = No mask

10 = Filter 15

01 = Acceptance Mask 1

00 = Acceptance Mask 0

bit 5-4 **FIL14\_<1:0>**: Filter 14 Select bits 1 and 0

11 = No mask

10 = Filter 15

01 = Acceptance Mask 1

00 = Acceptance Mask 0

bit 3-2 **FIL13\_<1:0>**: Filter 13 Select bits 1 and 0

11 = No mask

10 = Filter 15

01 = Acceptance Mask 1

00 = Acceptance Mask 0

bit 1-0 **FIL12\_<1:0>**: Filter 12 Select bits 1 and 0

11 = No mask

10 = Filter 15

01 = Acceptance Mask 1

00 = Acceptance Mask 0

**Note 1:** This register is available in Mode 1 and 2 only.



## 34.15.4 CAN BAUD RATE REGISTERS

This section describes the CAN Baud Rate registers.

**Note:** These registers are writable in Configuration mode only.

### REGISTER 34-52: BRGCON1: BAUD RATE CONTROL REGISTER 1

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| SJW1  | SJW0  | BRP5  | BRP4  | BRP3  | BRP2  | BRP1  | BRP0  |
| bit 7 |       |       |       |       |       |       | bit 0 |

#### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-6

**SJW<1:0>:** Synchronized Jump Width bits

11 = Synchronization jump width time = 4 x T<sub>Q</sub>

10 = Synchronization jump width time = 3 x T<sub>Q</sub>

01 = Synchronization jump width time = 2 x T<sub>Q</sub>

00 = Synchronization jump width time = 1 x T<sub>Q</sub>

bit 5-0

**BRP<5:0>:** Baud Rate Prescaler bits

111111 = T<sub>Q</sub> = (2 x 64)/F<sub>OSC</sub>

111110 = T<sub>Q</sub> = (2 x 63)/F<sub>OSC</sub>

⋮

⋮

000001 = T<sub>Q</sub> = (2 x 2)/F<sub>OSC</sub>

000000 = T<sub>Q</sub> = (2 x 1)/F<sub>OSC</sub>

## REGISTER 34-53: BRGCON2: BAUD RATE CONTROL REGISTER 2

|          |       |         |         |         |        |        |        |
|----------|-------|---------|---------|---------|--------|--------|--------|
| R/W-0    | R/W-0 | R/W-0   | R/W-0   | R/W-0   | R/W-0  | R/W-0  | R/W-0  |
| SEG2PHTS | SAM   | SEG1PH2 | SEG1PH1 | SEG1PH0 | PRSEG2 | PRSEG1 | PRSEG0 |
| bit 7    |       |         |         |         |        |        | bit 0  |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

- bit 7      **SEG2PHTS:** Phase Segment 2 Time Select bit  
1 = Freely programmable  
0 = Maximum of PHEG1 or Information Processing Time (IPT), whichever is greater
- bit 6      **SAM:** Sample of the CAN bus Line bit  
1 = Bus line is sampled three times prior to the sample point  
0 = Bus line is sampled once at the sample point
- bit 5-3    **SEG1PH<2:0>:** Phase Segment 1 bits  
111 = Phase Segment 1 time = 8 x T<sub>Q</sub>  
110 = Phase Segment 1 time = 7 x T<sub>Q</sub>  
101 = Phase Segment 1 time = 6 x T<sub>Q</sub>  
100 = Phase Segment 1 time = 5 x T<sub>Q</sub>  
011 = Phase Segment 1 time = 4 x T<sub>Q</sub>  
010 = Phase Segment 1 time = 3 x T<sub>Q</sub>  
001 = Phase Segment 1 time = 2 x T<sub>Q</sub>  
000 = Phase Segment 1 time = 1 x T<sub>Q</sub>
- bit 2-0    **PRSEG<2:0>:** Propagation Time Select bits  
111 = Propagation time = 8 x T<sub>Q</sub>  
110 = Propagation time = 7 x T<sub>Q</sub>  
101 = Propagation time = 6 x T<sub>Q</sub>  
100 = Propagation time = 5 x T<sub>Q</sub>  
011 = Propagation time = 4 x T<sub>Q</sub>  
010 = Propagation time = 3 x T<sub>Q</sub>  
001 = Propagation time = 2 x T<sub>Q</sub>  
000 = Propagation time = 1 x T<sub>Q</sub>

## REGISTER 34-54: BRGCON3: BAUD RATE CONTROL REGISTER 3

|        |        |     |     |     |                        |                        |                        |
|--------|--------|-----|-----|-----|------------------------|------------------------|------------------------|
| R/W-0  | R/W-0  | U-0 | U-0 | U-0 | R/W-0                  | R/W-0                  | R/W-0                  |
| WAKDIS | WAKFIL | —   | —   | —   | SEG2PH2 <sup>(1)</sup> | SEG2PH1 <sup>(1)</sup> | SEG2PH0 <sup>(1)</sup> |
| bit 7  |        |     |     |     |                        |                        | bit 0                  |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

- bit 7      **WAKDIS:** Wake-up Disable bit  
             1 = Disable CAN bus activity wake-up feature  
             0 = Enable CAN bus activity wake-up feature
- bit 6      **WAKFIL:** Selects CAN bus Line Filter for Wake-up bit  
             1 = Use CAN bus line filter for wake-up  
             0 = CAN bus line filter is not used for wake-up
- bit 5-3    **Unimplemented:** Read as '0'
- bit 2-0    **SEG2PH<2:0>:** Phase Segment 2 Time Select bits<sup>(1)</sup>  
             111 = Phase Segment 2 time = 8 x T<sub>Q</sub>  
             110 = Phase Segment 2 time = 7 x T<sub>Q</sub>  
             101 = Phase Segment 2 time = 6 x T<sub>Q</sub>  
             100 = Phase Segment 2 time = 5 x T<sub>Q</sub>  
             011 = Phase Segment 2 time = 4 x T<sub>Q</sub>  
             010 = Phase Segment 2 time = 3 x T<sub>Q</sub>  
             001 = Phase Segment 2 time = 2 x T<sub>Q</sub>  
             000 = Phase Segment 2 time = 1 x T<sub>Q</sub>

**Note 1:** These bits are ignored if SEG2PHTS bit (BRGCON2<7>) is '0'.

## 34.15.5 CAN MODULE I/O CONTROL REGISTER

This register controls the operation of the CAN module's I/O pins in relation to the rest of the microcontroller.

### REGISTER 34-55: CIOCON: CAN I/O CONTROL REGISTER

|        |     |     |     |     |     |     |        |
|--------|-----|-----|-----|-----|-----|-----|--------|
| R/W-0  | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0  |
| TX1SRC | —   | —   | —   | —   | —   | —   | CLKSEL |
| bit 7  |     |     |     |     |     |     | bit 0  |

#### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7 **TX1SRC:** CAN\_tx1 Signal Data Source bit

1 = CAN\_tx1 signal will output the CAN clock

0 = CAN\_tx1 signal will output CANTX

bit 6-1 **Unimplemented:** Read as '0'

bit 0 **CLKSEL:** CAN Clock Source Selection bit

1 = CAN clock is sourced by the clock selected by the FEXTOSC Configuration bit field, regardless of system clock<sup>(1)</sup>

0 = CAN clock is sourced from the system clock

**Note 1:** When CLKSEL = 1, the clock supplied by FEXTOSC must be less than or equal to the system clock. If the CAN clock is greater than the system clock, unexpected behavior will occur.

## 34.15.6 CAN INTERRUPT REGISTERS

The registers in this section are the same as described in [Section 9.0 “Interrupt Controller”](#). They are duplicated here for convenience.

### REGISTER 34-56: PIR5: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 5

| Mode 0   | R/W-0 | R/W-0 | R/W-0 | R/W-0  | R/W-0                 | R/W-0                 | R/W-0  | R/W-0    |
|----------|-------|-------|-------|--------|-----------------------|-----------------------|--------|----------|
|          | IRXIF | WAKIF | ERRIF | TXB2IF | TXB1IF <sup>(1)</sup> | TXB0IF <sup>(1)</sup> | RXB1IF | RXB0IF   |
| Mode 1,2 | R/W-0 | R/W-0 | R/W-0 | R/W-0  | R/W-0                 | R/W-0                 | R/W-0  | R/W-0    |
|          | IRXIF | WAKIF | ERRIF | TXBnIF | TXB1IF <sup>(1)</sup> | TXB0IF <sup>(1)</sup> | RXBnIF | FIFOWMIF |
| bit 7    |       |       |       |        |                       |                       |        | bit 0    |

#### Legend:

R = Readable bit                      W = Writable bit                      U = Unimplemented bit, read as ‘0’  
 -n = Value at POR                      ‘1’ = Bit is set                      ‘0’ = Bit is cleared                      x = Bit is unknown

- bit 7      **IRXIF**: CAN Bus Error Message Received Interrupt Flag bit  
             1 = An invalid message has occurred on the CAN bus  
             0 = No invalid message on the CAN bus
- bit 6      **WAKIF**: CAN Bus Activity Wake-up Interrupt Flag bit  
             1 = Activity on the CAN bus has occurred  
             0 = No activity on the CAN bus
- bit 5      **ERRIF**: CAN Module Error Interrupt Flag bit  
             1 = An error has occurred in the CAN module (multiple sources; refer to [Section 34.14.6 “Error Interrupt”](#))  
             0 = No CAN module errors
- bit 4      When CAN is in Mode 0:  
             **TXB2IF**: CAN Transmit Buffer 2 Interrupt Flag bit  
             1 = Transmit Buffer 2 has completed transmission of a message and may be reloaded  
             0 = Transmit Buffer 2 has not completed transmission of a message  
             When CAN is in Mode 1 or 2:  
             **TXBnIF**: Any Transmit Buffer Interrupt Flag bit  
             1 = One or more transmit buffers have completed transmission of a message and may be reloaded  
             0 = No transmit buffer is ready for reload
- bit 3      **TXB1IF**: CAN Transmit Buffer 1 Interrupt Flag bit<sup>(1)</sup>  
             1 = Transmit Buffer 1 has completed transmission of a message and may be reloaded  
             0 = Transmit Buffer 1 has not completed transmission of a message
- bit 2      **TXB0IF**: CAN Transmit Buffer 0 Interrupt Flag bit<sup>(1)</sup>  
             1 = Transmit Buffer 0 has completed transmission of a message and may be reloaded  
             0 = Transmit Buffer 0 has not completed transmission of a message
- bit 1      When CAN is in Mode 0:  
             **RXB1IF**: CAN Receive Buffer 1 Interrupt Flag bit  
             1 = Receive Buffer 1 has received a new message  
             0 = Receive Buffer 1 has not received a new message  
             When CAN is in Mode 1 or 2:  
             **RXBnIF**: Any Receive Buffer Interrupt Flag bit  
             1 = One or more receive buffers has received a new message  
             0 = No receive buffer has received a new message
- bit 0      When CAN is in Mode 0:  
             **RXB0IF**: CAN Receive Buffer 0 Interrupt Flag bit  
             1 = Receive Buffer 0 has received a new message  
             0 = Receive Buffer 0 has not received a new message  
             When CAN is in Mode 1:  
             **Unimplemented**: Read as ‘0’  
             When CAN is in Mode 2:  
             **FIFOWMIF**: FIFO Watermark Interrupt Flag bit  
             1 = FIFO high watermark is reached  
             0 = FIFO high watermark is not reached

**Note 1:** In CAN Mode 1 and 2, these bits are forced to ‘0’.

## REGISTER 34-57: PIE5: PERIPHERAL INTERRUPT ENABLE REGISTER 5

| Mode 0 | R/W-0 | R/W-0 | R/W-0 | R/W-0  | R/W-0                 | R/W-0                 | R/W-0  | R/W-0    |
|--------|-------|-------|-------|--------|-----------------------|-----------------------|--------|----------|
|        | IRXIE | WAKIE | ERRIE | TXB2IE | TXB1IE <sup>(1)</sup> | TXB0IE <sup>(1)</sup> | RXB1IE | RXB0IE   |
| Mode 1 | R/W-0 | R/W-0 | R/W-0 | R/W-0  | R/W-0                 | R/W-0                 | R/W-0  | R/W-0    |
|        | IRXIE | WAKIE | ERRIE | TXBnIE | TXB1IE <sup>(1)</sup> | TXB0IE <sup>(1)</sup> | RXBnIE | FIFOWMIE |
| bit 7  |       |       |       |        |                       |                       |        |          |
|        |       |       |       |        |                       |                       |        | bit 0    |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

- bit 7 **IRXIE:** CAN Bus Error Message Received Interrupt Enable bit  
1 = Enable invalid message received interrupt  
0 = Disable invalid message received interrupt
- bit 6 **WAKIE:** CAN bus Activity Wake-up Interrupt Enable bit  
1 = Enable bus activity wake-up interrupt  
0 = Disable bus activity wake-up interrupt
- bit 5 **ERRIE:** CAN bus Error Interrupt Enable bit  
1 = Enable CAN module error interrupt  
0 = Disable CAN module error interrupt
- bit 4 When CAN is in Mode 0:  
**TXB2IE:** CAN Transmit Buffer 2 Interrupt Enable bit  
1 = Enable Transmit Buffer 2 interrupt  
0 = Disable Transmit Buffer 2 interrupt  
When CAN is in Mode 1 or 2:  
**TXBnIE:** CAN Transmit Buffer Interrupts Enable bit  
1 = Enable transmit buffer interrupt; individual interrupt is enabled by TXBIE and BIE0  
0 = Disable all transmit buffer interrupts
- bit 3 **TXB1IE:** CAN Transmit Buffer 1 Interrupt Enable bit<sup>(1)</sup>  
1 = Enable Transmit Buffer 1 interrupt  
0 = Disable Transmit Buffer 1 interrupt
- bit 2 **TXB0IE:** CAN Transmit Buffer 0 Interrupt Enable bit<sup>(1)</sup>  
1 = Enable Transmit Buffer 0 interrupt  
0 = Disable Transmit Buffer 0 interrupt
- bit 1 When CAN is in Mode 0:  
**RXB1IE:** CAN Receive Buffer 1 Interrupt Enable bit  
1 = Enable Receive Buffer 1 interrupt  
0 = Disable Receive Buffer 1 interrupt  
When CAN is in Mode 1 or 2:  
**RXBnIE:** CAN Receive Buffer Interrupts Enable bit  
1 = Enable receive buffer interrupt; individual interrupt is enabled by BIE0  
0 = Disable all receive buffer interrupts
- bit 0 When CAN is in Mode 0:  
**RXB0IE:** CAN Receive Buffer 0 Interrupt Enable bit  
1 = Enable Receive Buffer 0 interrupt  
0 = Disable Receive Buffer 0 interrupt  
When CAN is in Mode 1:  
**Unimplemented:** Read as '0'  
When CAN is in Mode 2:  
**FIFOWMIE:** FIFO Watermark Interrupt Enable bit  
1 = Enable FIFO watermark interrupt  
0 = Disable FIFO watermark interrupt

**Note 1:** In CAN Mode 1 and 2, these bits are forced to '0'.

## REGISTER 34-58: IPR5: PERIPHERAL INTERRUPT PRIORITY REGISTER 5

|                 |       |       |       |        |                       |                       |        |          |
|-----------------|-------|-------|-------|--------|-----------------------|-----------------------|--------|----------|
| <b>Mode 0</b>   | R/W-1 | R/W-1 | R/W-1 | R/W-1  | R/W-1                 | R/W-1                 | R/W-1  | R/W-1    |
|                 | IRXIP | WAKIP | ERRIP | TXB2IP | TXB1IP <sup>(1)</sup> | TXB0IP <sup>(1)</sup> | RXB1IP | RXB0IP   |
| <b>Mode 1,2</b> | R/W-1 | R/W-1 | R/W-1 | R/W-1  | R/W-1                 | R/W-1                 | R/W-1  | R/W-1    |
|                 | IRXIP | WAKIP | ERRIP | TXBnIP | TXB1IP <sup>(1)</sup> | TXB0IP <sup>(1)</sup> | RXBnIP | FIFOWMIP |
| bit 7           |       |       |       |        |                       |                       |        | bit 0    |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7 **IRXIP:** CAN Bus Error Message Received Interrupt Priority bit

1 = High priority

0 = Low priority

bit 6 **WAKIP:** CAN Bus Activity Wake-up Interrupt Priority bit

1 = High priority

0 = Low priority

bit 5 **ERRIP:** CAN Module Error Interrupt Priority bit

1 = High priority

0 = Low priority

bit 4 When CAN is in Mode 0:

**TXB2IP:** CAN Transmit Buffer 2 Interrupt Priority bit

1 = High priority

0 = Low priority

When CAN is in Mode 1 or 2:

**TXBnIP:** CAN Transmit Buffer Interrupt Priority bit

1 = High priority

0 = Low priority

bit 3 **TXB1IP:** CAN Transmit Buffer 1 Interrupt Priority bit<sup>(1)</sup>

1 = High priority

0 = Low priority

bit 2 **TXB0IP:** CAN Transmit Buffer 0 Interrupt Priority bit<sup>(1)</sup>

1 = High priority

0 = Low priority

bit 1 When CAN is in Mode 0:

**RXB1IP:** CAN Receive Buffer 1 Interrupt Priority bit

1 = High priority

0 = Low priority

When CAN is in Mode 1 or 2:

**RXBnIP:** CAN Receive Buffer Interrupts Priority bit

1 = High priority

0 = Low priority

**Note 1:** In CAN Mode 1 and 2, these bits are forced to '0'.

## REGISTER 34-58: IPR5: PERIPHERAL INTERRUPT PRIORITY REGISTER 5 (CONTINUED)

bit 0      When CAN is in Mode 0:  
**RXB0IP:** CAN Receive Buffer 0 Interrupt Priority bit  
 1 = High priority  
 0 = Low priority  
When CAN is in Mode 1:  
**Unimplemented:** Read as '0'  
When CAN is in Mode 2:  
**FIFOWMIP:** FIFO Watermark Interrupt Priority bit  
 1 = High priority  
 0 = Low priority

**Note 1:** In CAN Mode 1 and 2, these bits are forced to '0'.

## REGISTER 34-59: TXBIE: TRANSMIT BUFFERS INTERRUPT ENABLE REGISTER<sup>(1)</sup>

|       |     |     |                       |                       |                       |     |       |
|-------|-----|-----|-----------------------|-----------------------|-----------------------|-----|-------|
| U-0   | U-0 | U-0 | R/W-0                 | R/W-0                 | R/W-0                 | U-0 | U-0   |
| —     | —   | —   | TXB2IE <sup>(2)</sup> | TXB1IE <sup>(2)</sup> | TXB0IE <sup>(2)</sup> | —   | —     |
| bit 7 |     |     |                       |                       |                       |     |       |
|       |     |     |                       |                       |                       |     | bit 0 |

### Legend:

R = Readable bit      W = Writable bit      U = Unimplemented bit, read as '0'  
 -n = Value at POR      '1' = Bit is set      '0' = Bit is cleared      x = Bit is unknown

bit 7-5      **Unimplemented:** Read as '0'  
 bit 4-2      **TXB2IE:TXB0IE:** Transmit Buffer 2-0 Interrupt Enable bits<sup>(2)</sup>  
 1 = Transmit buffer interrupt is enabled  
 0 = Transmit buffer interrupt is disabled  
 bit 1-0      **Unimplemented:** Read as '0'

**Note 1:** This register is available in Mode 1 and 2 only.  
**2:** TXBnIE in PIE5 register must be set to get an interrupt.



## REGISTER 34-60: BIE0: BUFFER INTERRUPT ENABLE REGISTER 0<sup>(1)</sup>

|                     |                     |                     |                     |                     |                     |                       |                       |
|---------------------|---------------------|---------------------|---------------------|---------------------|---------------------|-----------------------|-----------------------|
| R/W-0               | R/W-0               | R/W-0               | R/W-0               | R/W-0               | R/W-0               | R/W-0                 | R/W-0                 |
| B5IE <sup>(2)</sup> | B4IE <sup>(2)</sup> | B3IE <sup>(2)</sup> | B2IE <sup>(2)</sup> | B1IE <sup>(2)</sup> | B0IE <sup>(2)</sup> | RXB1IE <sup>(2)</sup> | RXB0IE <sup>(2)</sup> |
| bit 7               |                     |                     |                     |                     |                     | bit 0                 |                       |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-2 **B<5:0>IE:** Programmable Transmit/Receive Buffer 5-0 Interrupt Enable bits<sup>(2)</sup>

1 = Interrupt is enabled

0 = Interrupt is disabled

bit 1-0 **RXB<1:0>IE:** Dedicated Receive Buffer 1-0 Interrupt Enable bits<sup>(2)</sup>

1 = Interrupt is enabled

0 = Interrupt is disabled

**Note 1:** This register is available in Mode 1 and 2 only.

**2:** Either TXBnIE or RXBnIE, in the PIE5 register, must be set to get an interrupt.

## 35.0 FIXED VOLTAGE REFERENCE (FVR)

The Fixed Voltage Reference, or FVR, is a stable voltage reference, independent of  $V_{DD}$ , with 1.024V, 2.048V or 4.096V selectable output levels. The output of the FVR can be configured to supply a reference voltage to the following:

- ADC input channel
- ADC positive reference
- Comparator input
- Digital-to-Analog Converter (DAC)

The FVR can be enabled by setting the EN bit of the FVRCON register.

**Note:** Fixed Voltage Reference output cannot exceed  $V_{DD}$ .

### 35.1 Independent Gain Amplifiers

The output of the FVR, which is connected to the ADC, Comparators, and DAC, is routed through two independent programmable gain amplifiers. Each amplifier can be programmed for a gain of 1x, 2x or 4x, to produce the three possible voltage levels.

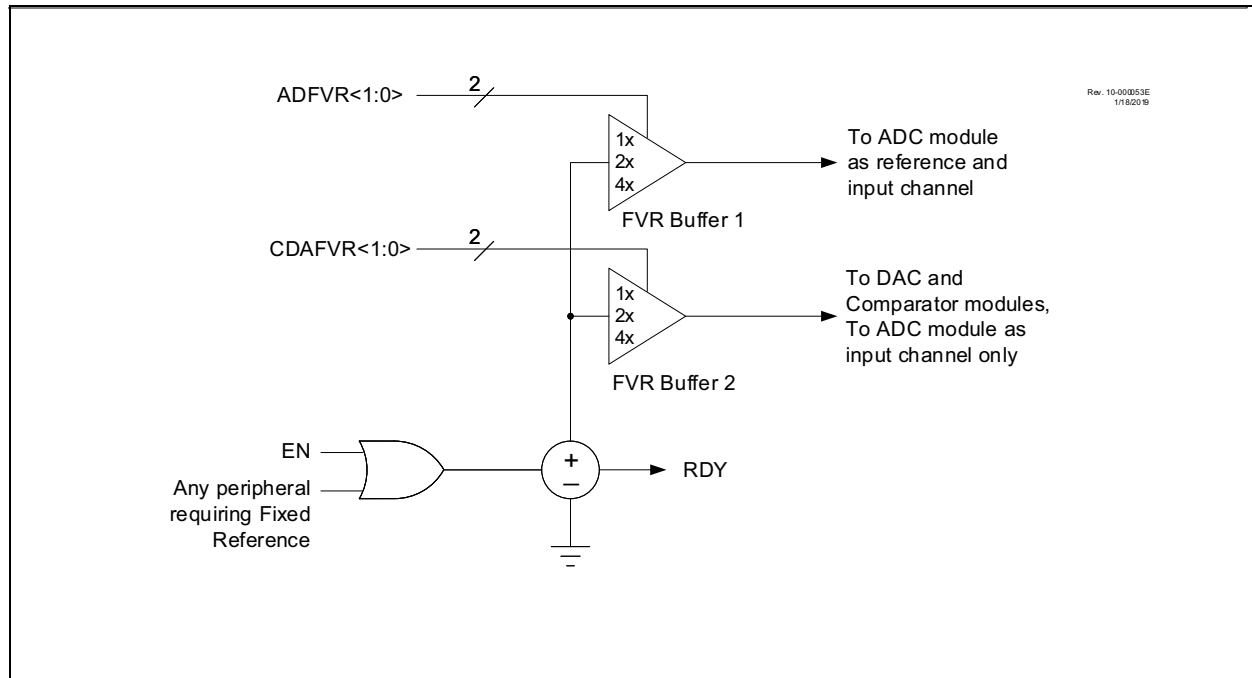
The ADFVR<1:0> bits of the FVRCON register are used to enable and configure the gain amplifier settings for the reference supplied to the ADC module. Reference [Section 37.0 “Analog-to-Digital Converter with Computation \(ADC2\) Module”](#) for additional information.

The CDAFVR<1:0> bits of the FVRCON register are used to enable and configure the gain amplifier settings for the reference supplied to the DAC and comparator module. Reference [Section 38.0 “5-Bit Digital-to-Analog Converter \(DAC\) Module”](#) and [Section 39.0 “Comparator Module”](#) for additional information.

### 35.2 FVR Stabilization Period

When the Fixed Voltage Reference module is enabled, it requires time for the reference and amplifier circuits to stabilize. Once the circuits stabilize and are ready for use, the RDY bit of the FVRCON register will be set.

**FIGURE 35-1: VOLTAGE REFERENCE BLOCK DIAGRAM**



## 35.3 Register Definitions: FVR Control

**REGISTER 35-1: FVRCON: FIXED VOLTAGE REFERENCE CONTROL REGISTER**

|         |       |                     |                      |             |         |            |         |
|---------|-------|---------------------|----------------------|-------------|---------|------------|---------|
| R/W-0/0 | R-q/q | R/W-0/0             | R/W-0/0              | R/W-0/0     | R/W-0/0 | R/W-0/0    | R/W-0/0 |
| EN      | RDY   | TSEN <sup>(2)</sup> | TSRNG <sup>(2)</sup> | CDAFVR<1:0> |         | ADFVR<1:0> |         |
| bit 7   |       |                     |                      |             |         |            | bit 0   |

**Legend:**

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared | q = Value depends on condition                        |

|         |                                                                                                                                                                                                                                                             |
|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| bit 7   | <b>EN:</b> Fixed Voltage Reference Enable bit<br>1 = Fixed Voltage Reference is enabled<br>0 = Fixed Voltage Reference is disabled                                                                                                                          |
| bit 6   | <b>RDY:</b> Fixed Voltage Reference Ready Flag bit<br>1 = Fixed Voltage Reference output is ready for use<br>0 = Fixed Voltage Reference output is not ready or not enabled                                                                                 |
| bit 5   | <b>TSEN:</b> Temperature Indicator Enable bit <sup>(2)</sup><br>1 = Temperature Indicator is enabled<br>0 = Temperature Indicator is disabled                                                                                                               |
| bit 4   | <b>TSRNG:</b> Temperature Indicator Range Selection bit <sup>(2)</sup><br>1 = V <sub>OUT</sub> = 3V <sub>T</sub> (High Range)<br>0 = V <sub>OUT</sub> = 2V <sub>T</sub> (Low Range)                                                                         |
| bit 3-2 | <b>CDAFVR&lt;1:0&gt;:</b> Comparator FVR Buffer Gain Selection bits<br>11 = FVR Buffer 2 Gain is 4x, (4.096V) <sup>(1)</sup><br>10 = FVR Buffer 2 Gain is 2x, (2.048V) <sup>(1)</sup><br>01 = FVR Buffer 2 Gain is 1x, (1.024V)<br>00 = FVR Buffer 2 is off |
| bit 1-0 | <b>ADFVR&lt;1:0&gt;:</b> ADC FVR Buffer Gain Selection bit<br>11 = FVR Buffer 1 Gain is 4x, (4.096V) <sup>(1)</sup><br>10 = FVR Buffer 1 Gain is 2x, (2.048V) <sup>(1)</sup><br>01 = FVR Buffer 1 Gain is 1x, (1.024V)<br>00 = FVR Buffer 1 is off          |

**Note 1:** Fixed Voltage Reference output cannot exceed V<sub>DD</sub>.

**2:** See [Section 36.0 “Temperature Indicator Module”](#) for additional information.

**TABLE 35-1: SUMMARY OF REGISTERS ASSOCIATED WITH FIXED VOLTAGE REFERENCE**

| Name   | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3       | Bit 2 | Bit 1      | Bit 0 | Register on page    |
|--------|-------|-------|-------|-------|-------------|-------|------------|-------|---------------------|
| FVRCON | EN    | RDY   | TSEN  | TSRNG | CDAFVR<1:0> |       | ADFVR<1:0> |       | <a href="#">651</a> |

**Legend:** — = Unimplemented location, read as '0'. Shaded cells are not used with the Fixed Voltage Reference.

## 36.0 TEMPERATURE INDICATOR MODULE

This family of devices is equipped with a temperature circuit designed to measure the operating temperature of the silicon die.

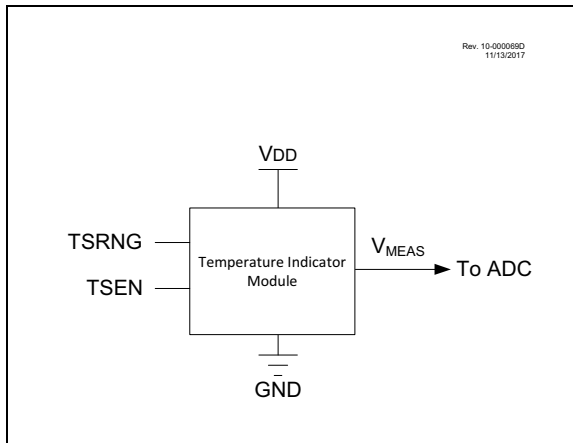
The circuit's range of operating temperature falls between  $-40^{\circ}\text{C}$  and  $+125^{\circ}\text{C}$ . A one-point calibration allows the circuit to indicate a temperature closely surrounding that point. A two-point calibration allows the circuit to sense the entire range of temperature more accurately.

### 36.1 Module Operation

The temperature indicator module consists of a temperature-sensing circuit that provides a voltage to the device ADC. The analog voltage output,  $V_{\text{MEAS}}$ , varies inversely to the device temperature. The output of the temperature indicator is referred to as  $V_{\text{MEAS}}$ .

Figure 36-1 shows a simplified block diagram of the temperature indicator module.

**FIGURE 36-1: TEMPERATURE INDICATOR MODULE BLOCK DIAGRAM**



The output of the circuit is measured using the internal Analog-to-Digital Converter. A channel is reserved for the temperature circuit output. Refer to [Section 37.0 “Analog-to-Digital Converter with Computation \(ADC2\) Module”](#) for detailed information.

The ON/OFF bit for the module is located in the FVRCON register. See [Section 35.0 “Fixed Voltage Reference \(FVR\)”](#) for more information. The circuit is enabled by setting the TSEN bit of the FVRCON register. When the module is disabled, the circuit draws no current.

The circuit operates in either High or Low range. Refer to [Section 36.1.2 “Temperature Indicator Range”](#) for more details on the range settings.

#### 36.1.1 MINIMUM OPERATING $V_{\text{DD}}$

When the temperature circuit is operated in low range, the device may be operated at any operating voltage that is within specifications. When the temperature circuit is operated in high range, the device operating voltage,  $V_{\text{DD}}$ , must be high enough to ensure that the temperature circuit is correctly biased.

Table 36-1 shows the recommended minimum  $V_{\text{DD}}$  vs. Range setting.

**TABLE 36-1: RECOMMENDED  $V_{\text{DD}}$  vs. RANGE**

| Min. $V_{\text{DD}}$ , TSRNG = 1<br>(High Range) | Min. $V_{\text{DD}}$ , TSRNG = 0<br>(Low Range) |
|--------------------------------------------------|-------------------------------------------------|
| $\geq 2.5$                                       | $\geq 1.8$                                      |

#### 36.1.2 TEMPERATURE INDICATOR RANGE

The temperature indicator circuit operates in either high or low range. The high range, selected by setting the TSRNG bit of the FVRCON register, provides a wider output voltage. This provides more resolution over the temperature range. High range requires a higher-bias voltage to operate and thus, a higher  $V_{\text{DD}}$  is needed. The low range is selected by clearing the TSRNG bit of the FVRCON register. The low range generates a lower sensor voltage and thus, a lower  $V_{\text{DD}}$  voltage is needed to operate the circuit.

The output voltage of the sensor is the highest value at  $-40^{\circ}\text{C}$  and the lowest value at  $+125^{\circ}\text{C}$ .

- **High Range:** The High range is used in applications with the reference for the ADC,  $V_{\text{REF}} = 2.048\text{V}$ . This range may not be suitable for battery-powered applications.
- **Low Range:** This mode is useful in applications in which the  $V_{\text{DD}}$  is too low for high-range operation. The  $V_{\text{DD}}$  in this mode can be as low as 1.8V.  $V_{\text{DD}}$  must, however, be at least 0.5V higher than the maximum sensor voltage, depending on the expected low operating temperature.

## 36.2 Temperature Calculation

This section describes the steps involved in estimating the die temperature, TMEAS:

1. Obtain the ADC count value of the measured analog voltage: The analog output voltage, VMEAS is converted to a digital count value by the Analog-to-Digital Converter (ADC) and is referred to as ADCMEAS.
2. Obtain the ADC count value, ADCDIA at 90 degrees, from the DIA table. This parameter is TSLR2 for the low range setting or TSHR2 for the high range setting of the temperature indicator module.
3. Obtain the output analog voltage (in mV) value of the Fixed Reference Voltage (FVR) for 2x setting, from the DIA Table. This parameter is FVRA2X in the DIA table (Table 5-3).
4. Obtain the value of the temperature indicator voltage sensitivity, parameter Mv, from Table 45-25 for the corresponding range setting.

Equation 36-1 provides an estimate for the die temperature based on the above parameters.

### EQUATION 36-1: SENSOR TEMPERATURE

$$T_{MEAS} = 90 + \frac{(ADC_{MEAS} - ADC_{DIA}) \times FVRA2X}{(2^N - 1) \times M_V}$$

Where:

ADCMEAS = ADC reading at temperature being estimated

ADCDIA = ADC reading stored in the DIA

FVRA2X = FVR value stored in the DIA for 2x setting

N = Resolution of the ADC

Mv = Temperature Indicator voltage sensitivity (mV/°C)

**Note:** It is recommended to take the average of ten measurements of ADCMEAS to reduce noise and improve accuracy.

### 36.2.1 CALIBRATION

#### 36.2.1.1 Higher-Order Calibration

If the application requires more precise temperature measurement, additional calibrations steps will be necessary. For these applications, two-point or three-point calibration is recommended.

### 36.2.2 TEMPERATURE RESOLUTION

The resolution of the ADC reading, Ma (°C/count), depends on both the ADC resolution N and the reference voltage used for conversion, as shown in Equation 36-1. It is recommended to use the smallest VREF value, such as the ADC FVR1 output voltage for 2x setting (FVRA2X) value from the DIA, instead of VDD. Refer to Table 5-3 for DIA location.

**Note:** Refer to Table 45-17 for FVR reference voltage accuracy.

## 36.3 ADC Acquisition Time

To ensure accurate temperature measurements, the user must wait a certain minimum acquisition time (Table 45-25) for the ADC value to settle, after the ADC input multiplexer is connected to the temperature indicator output, before the conversion is performed.

**TABLE 36-2: SUMMARY OF REGISTERS ASSOCIATED WITH THE TEMPERATURE INDICATOR**

| Name   | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3       | Bit 2 | Bit 1      | Bit 0 | Register on page |
|--------|-------|-------|-------|-------|-------------|-------|------------|-------|------------------|
| FVRCON | EN    | RDY   | TSEN  | TSRNG | CDAFVR<1:0> |       | ADFVR<1:0> |       | 651              |

**Legend:** — = Unimplemented location, read as '0'. Shaded cells are unused by the temperature indicator module.

## 37.0 ANALOG-TO-DIGITAL CONVERTER WITH COMPUTATION (ADC<sup>2</sup>) MODULE

The Analog-to-Digital Converter with Computation (ADC<sup>2</sup>) allows conversion of an analog input signal to a 12-bit binary representation of that signal. This device uses analog inputs, which are multiplexed into a single sample and hold circuit. The output of the sample and hold is connected to the input of the converter. The converter generates a 12-bit binary result via successive approximation and stores the conversion result into the ADC result registers (ADRESH:ADRESL register pair).

Additionally, the following features are provided within the ADC module:

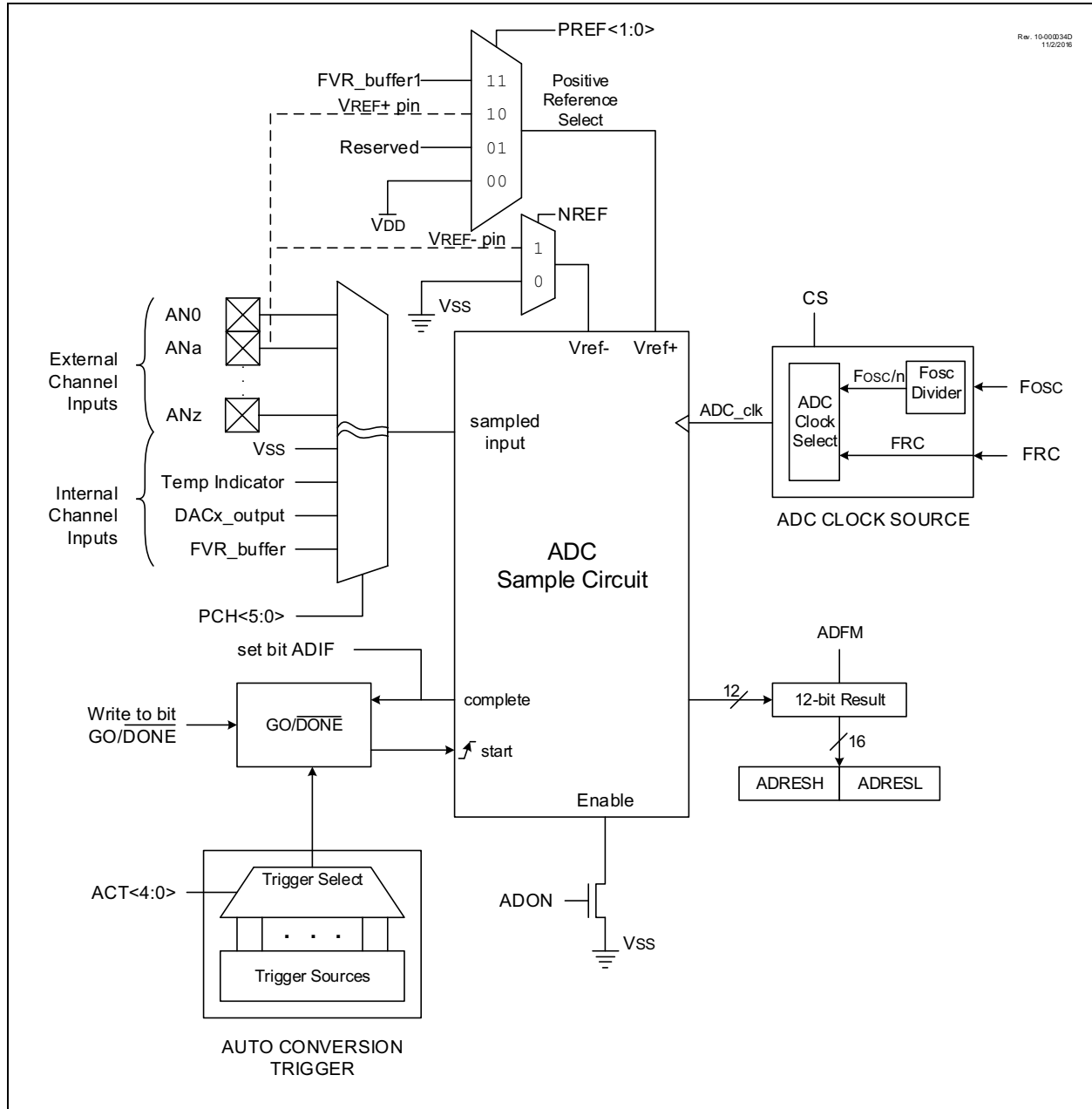
- 13-bit Acquisition Timer
- Hardware Capacitive Voltage Divider (CVD) support:
  - 13-bit Precharge Timer
  - Adjustable sample and hold capacitor array
  - Guard ring digital output drive
- Automatic repeat and sequencing:
  - Automated double sample conversion for CVD
  - Two sets of result registers (Result and Previous result)
  - Auto-conversion trigger
  - Internal retrigger
- Computation features:
  - Averaging and Low-Pass Filter functions
  - Reference Comparison
  - 2-level Threshold Comparison
  - Selectable Interrupts

Figure 37-1 shows the block diagram of the ADC.

The ADC voltage reference is software selectable to be either internally generated or externally supplied.

The ADC can generate an interrupt upon completion of a conversion and upon threshold comparison. These interrupts can be used to wake up the device from Sleep.

**FIGURE 37-1: ADC<sup>2</sup> BLOCK DIAGRAM**



## 37.1 ADC Configuration

When configuring and using the ADC the following functions must be considered:

- Port configuration
- Channel selection
- ADC voltage reference selection
- ADC conversion clock source
- Interrupt control
- Result formatting
- Conversion Trigger Selection
- ADC Acquisition Time
- ADC Precharge Time
- Additional Sample and Hold Capacitor
- Single/Double Sample Conversion
- Guard Ring Outputs

### 37.1.1 PORT CONFIGURATION

The ADC can be used to convert both analog and digital signals. When converting analog signals, the I/O pin should be configured for analog by setting the associated TRIS and ANSEL bits. Refer to [Section 16.0 “I/O Ports”](#) for more information.

**Note:** Analog voltages on any pin that is defined as a digital input may cause the input buffer to conduct excess current.

### 37.1.2 CHANNEL SELECTION

There are several channel selections available:

- Eight PORTA pins (RA<7:0>)
- Eight PORTB pins (RB<7:0>)
- Eight PORTC pins (RC<7:0>)
- Temperature Indicator
- DAC output
- Fixed Voltage Reference (FVR)
- Vss (ground)

The ADPCH register determines which channel is connected to the sample and hold circuit.

When changing channels, a delay is required before starting the next conversion.

Refer to [Section 37.2 “ADC Operation”](#) for more information.

### 37.1.3 ADC VOLTAGE REFERENCE

The ADPREF<1:0> bits of the ADREF register provide control of the positive voltage reference. The positive voltage reference can be:

- VREF+ pin
- VDD
- FVR outputs

The ADNREF bit of the ADREF register provides control of the negative voltage reference. The negative voltage reference can be:

- VREF- pin
- VSS

See [Section 35.0 “Fixed Voltage Reference \(FVR\)”](#) for more details on the Fixed Voltage Reference.

### 37.1.4 CONVERSION CLOCK

The source of the conversion clock is software selectable via the ADCLK register and the CS bits of the ADCON0 register. If FOSC is selected as the ADC clock, there is a prescaler available to divide the clock so that it meets the ADC clock period specification. The ADC clock source options are the following:

- FOSC/(2<sup>n</sup>)(where n is from 1 to 128)
- FRC (dedicated RC oscillator)

The time to complete one bit conversion is defined as TAD. Refer [Figure 37-2](#) for the complete timing details of the ADC conversion.

For correct conversion, the appropriate TAD specification must be met. Refer to [Table 45-14](#) for more information. [Table 37-1](#) gives examples of appropriate ADC clock selections.

**Note 1:** Unless using the FRC, any changes in the system clock frequency will change the ADC clock frequency, which may adversely affect the ADC result.

**2:** The internal control logic of the ADC runs off of the clock selected by the CS bit of ADCON0. What this can mean is when the CS bit of ADCON0 is set to ‘1’ (ADC runs on FRC), there may be unexpected delays in operation when setting ADC control bits.



**TABLE 37-1: ADC CLOCK PERIOD (T<sub>AD</sub>) Vs. DEVICE OPERATING FREQUENCIES<sup>(1,4)</sup>**

| ADC Clock Period (T <sub>AD</sub> ) |                   | Device Frequency (F <sub>osc</sub> ) |                         |                       |                       |                        |                        |                         |
|-------------------------------------|-------------------|--------------------------------------|-------------------------|-----------------------|-----------------------|------------------------|------------------------|-------------------------|
| ADC Clock Source                    | CS<5:0>           | 64 MHz                               | 32 MHz                  | 20 MHz                | 16 MHz                | 8 MHz                  | 4 MHz                  | 1 MHz                   |
| FOSC/2                              | 000000            | 31.25 ns <sup>(2)</sup>              | 62.5 ns <sup>(2)</sup>  | 100 ns <sup>(2)</sup> | 125 ns <sup>(2)</sup> | 250 ns <sup>(2)</sup>  | 500 ns                 | 2.0 μs                  |
| FOSC/4                              | 000001            | 62.5 ns <sup>(2)</sup>               | 125 ns <sup>(2)</sup>   | 200 ns <sup>(2)</sup> | 250 ns <sup>(2)</sup> | 500 ns                 | 1.0 μs                 | 4.0 μs                  |
| FOSC/6                              | 000010            | 125 ns <sup>(2)</sup>                | 187.5 ns <sup>(2)</sup> | 300 ns <sup>(2)</sup> | 375 ns <sup>(2)</sup> | 750 ns                 | 1.5 μs                 | 6.0 μs                  |
| FOSC/8                              | 000011            | 187.5 ns <sup>(2)</sup>              | 250 ns <sup>(2)</sup>   | 400 ns <sup>(2)</sup> | 500 ns                | 1.0 μs                 | 2.0 μs                 | 8.0 μs                  |
| ...                                 | ...               | ...                                  | ...                     | ...                   | ...                   | ...                    | ...                    | ...                     |
| FOSC/16                             | 000111            | 250 ns <sup>(2)</sup>                | 500 ns                  | 800 ns                | 1.0 μs                | 2.0 μs                 | 4.0 μs                 | 16.0 μs <sup>(3)</sup>  |
| ...                                 | ...               | ...                                  | ...                     | ...                   | ...                   | ...                    | ...                    | ...                     |
| FOSC/128                            | 111111            | 2.0 μs                               | 4.0 μs                  | 6.4 μs                | 8.0 μs                | 16.0 μs <sup>(3)</sup> | 32.0 μs <sup>(2)</sup> | 128.0 μs <sup>(2)</sup> |
| FRC                                 | CS(ADCON0<4>) = 1 | 1.0-6.0 μs                           | 1.0-6.0 μs              | 1.0-6.0 μs            | 1.0-6.0 μs            | 1.0-6.0 μs             | 1.0-6.0 μs             | 1.0-6.0 μs              |

**Legend:** Shaded cells are outside of recommended range.

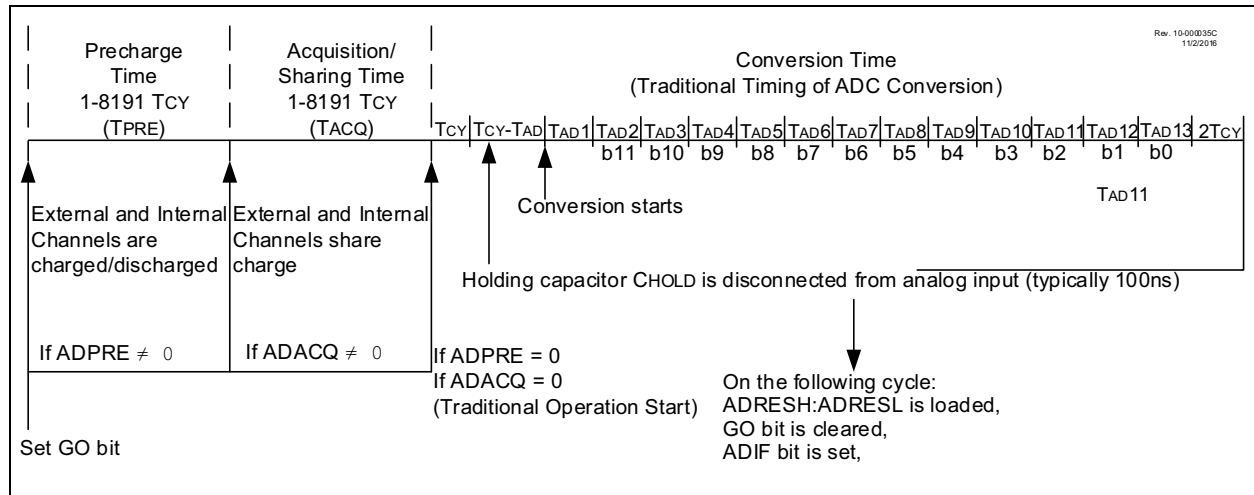
**Note 1:** See TAD parameter for FRC source typical TAD value.

**2:** These values violate the required TAD time.

**3:** Outside the recommended TAD time.

**4:** The ADC clock period (T<sub>AD</sub>) and total ADC conversion time can be minimized when the ADC clock is derived from the system clock FOSC. However, the FRC oscillator source must be used when conversions are to be performed with the device in Sleep mode.

**FIGURE 37-2: ANALOG-TO-DIGITAL CONVERSION TAD CYCLES**



## 37.1.5 INTERRUPTS

The ADC module allows for the ability to generate an interrupt upon completion of an Analog-to-Digital conversion. The ADC Interrupt Flag is the ADIF bit in the PIR1 register. The ADC Interrupt Enable is the ADIE bit in the PIE1 register. The ADIF bit must be cleared in software.

- Note 1:** The ADIF bit is set at the completion of every conversion, regardless of whether or not the ADC interrupt is enabled.
- 2:** The ADC operates during Sleep only when the FRC oscillator is selected.

This interrupt can be generated while the device is operating or while in Sleep. If the device is in Sleep, the interrupt will wake up the device. Upon waking from Sleep, the next instruction following the `SLEEP` instruction is always executed. If the user is attempting to wake up from Sleep and resume in-line code execution, the ADIE bit of the PIE<sub>x</sub> register and the GIE bits of the INTCON0 register must both be set. If all these bits are set, the execution will switch to the Interrupt Service Routine.

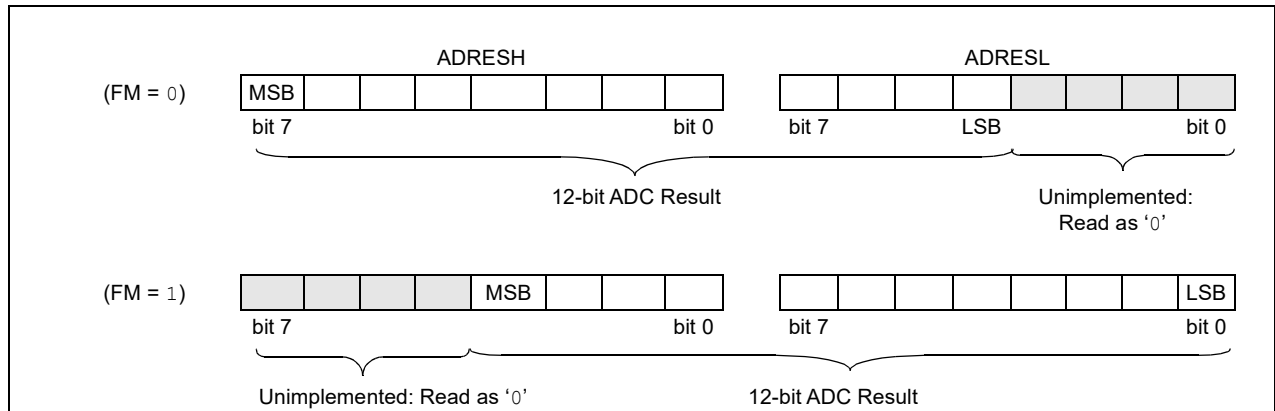
## 37.1.6 RESULT FORMATTING

The 12-bit ADC conversion result can be supplied in two formats, left justified or right justified. The FM bits of the ADCON0 register controls the output format.

Figure 37-3 shows the two output formats.

Writes to the ADRES register pair are always right justified regardless of the selected format mode. Therefore, data read after writing to ADRES when ADFRM0 = 0 will be shifted left four places.

**FIGURE 37-3: 12-BIT ADC CONVERSION RESULT FORMAT**



## 37.2 ADC Operation

### 37.2.1 STARTING A CONVERSION

To enable the ADC module, the ON bit of the ADCON0 register must be set to a '1'. A conversion may be started by any of the following:

- Software setting the GO bit of ADCON0 to '1'
- An external trigger (selected by [Register 37-3](#))
- A continuous-mode retrigger (see section [Section 37.6.8 "Continuous Sampling mode"](#))

**Note:** The GO bit should not be set in the same instruction that turns on the ADC. Refer to [Section 37.2.6 "ADC Conversion Procedure \(Basic Mode\)"](#).

### 37.2.2 COMPLETION OF A CONVERSION

When any individual conversion is complete, the value already in ADRES is written into PREV (if ADPSIS = 1) and the new conversion results appear in ADRES. When the conversion completes, the ADC module will:

- Clear the GO bit (unless the CONT bit of ADCON0 is set)
- Set the ADIF Interrupt Flag bit
- Set the ADMATH bit
- Update ACC

When ADDSEN = 0 then after every conversion, or when ADDSEN = 1 then after every other conversion, the following events occur:

- ERR is calculated
- ADTIF is set if ERR calculation meets threshold comparison

Importantly, filter and threshold computations occur after the conversion itself is complete. As such, interrupt handlers responding to ADIF should check ADTIF before reading filter and threshold results.

### 37.2.3 ADC OPERATION DURING SLEEP

The ADC module can operate during Sleep. This requires the ADC clock source to be set to the FRC option. When the FRC oscillator source is selected, the ADC waits one additional instruction before starting the conversion. This allows the SLEEP instruction to be executed, which can reduce system noise during the conversion. If the ADC interrupt is enabled, the device will wake-up from Sleep when the conversion completes. If the ADC interrupt is disabled, the ADC module is turned off after the conversion completes, although the ON bit remains set.

### 37.2.4 EXTERNAL TRIGGER DURING SLEEP

If the external trigger is received during Sleep while ADC clock source is set to the FRC, ADC module will perform the conversion and set the ADIF bit upon completion.

If an external trigger is received when the ADC clock source is something other than FRC, the trigger will be recorded, but the conversion will not begin until the device exits Sleep.

## 37.2.5 AUTO-CONVERSION TRIGGER

The auto-conversion trigger allows periodic ADC measurements without software intervention. When a rising edge of the selected source occurs, the GO bit is set by hardware.

The auto-conversion trigger source is selected by the ADACT register.

Using the auto-conversion trigger does not assure proper ADC timing. It is the user's responsibility to ensure that the ADC timing requirements are met. See [Register 37-33](#) for auto-conversion sources.

## 37.2.6 ADC CONVERSION PROCEDURE (BASIC MODE)

This is an example procedure for using the ADC to perform an analog-to-digital conversion:

1. Configure Port:
  - Disable pin output driver (Refer to the TRISx register)
  - Configure pin as analog (Refer to the ANSELx register)
2. Configure the ADC module:
  - Select ADC conversion clock
  - Select voltage reference
  - Select ADC input channel

- Precharge and acquisition
  - Turn on ADC module
3. Configure ADC interrupt (optional):
    - Clear ADC interrupt flag
    - Enable ADC interrupt
    - Enable global interrupt (GIEL bit)<sup>(1)</sup>
  4. If ADACQ = 0, software must wait the required acquisition time<sup>(2)</sup>.
  5. Start conversion by setting the GO bit.
  6. Wait for ADC conversion to complete by one of the following:
    - Polling the GO bit
    - Polling the ADIF bit
    - Waiting for the ADC interrupt (interrupts enabled)
  7. Read ADC Result.
  8. Clear the ADC interrupt flag (required if interrupt is enabled).

**Note 1:** The global interrupt can be disabled if the user is attempting to wake-up from Sleep and resume in-line code execution.

**2:** Refer to [Section 37.3 “ADC Acquisition Requirements”](#).

### EXAMPLE 37-1: ADC CONVERSION

```
/*This code block configures the ADC
for polling, VDD and VSS references, FRC
oscillator and AN0 input.
Conversion start & polling for completion
are included.
*/
void main() {
    //System Initialize
    initializeSystem();

    //Setup ADC
    ADCON0bits.FM = 1; //right justify
    ADCON0bits.CS = 1; //FRC Clock
    ADPCH = 0x00; //RA0 is Analog channel
    TRISAbits.TRISA0 = 1; //Set RA0 to input
    ANSELAbits.ANSELA0 = 1; //Set RA0 to analog
    ADCON0bits.ON = 1; //Turn ADC On

    while (1) {
        ADCON0bits.GO = 1; //Start conversion
        while (ADCON0bits.GO); //Wait for conversion done
        resultHigh = ADRESH; //Read result
        resultLow = ADRESL; //Read result
    }
}
```

## 37.3 ADC Acquisition Requirements

For the ADC to meet its specified accuracy, the charge holding capacitor (CHOLD) must be allowed to fully charge to the input channel voltage level. The Analog Input model is shown in [Figure 37-4](#). The source impedance (Rs) and the internal sampling switch (RSS) impedance directly affect the time required to charge the capacitor CHOLD. The sampling switch (RSS) impedance varies over the device voltage (VDD), refer to [Figure 37-4](#). **The maximum recommended impedance for analog sources is 10 kΩ.** If the source

impedance is decreased, the acquisition time may be decreased. After the analog input channel is selected (or changed), an ADC acquisition must be completed before the conversion can be started. To calculate the minimum acquisition time, [Equation 37-1](#) may be used. This equation assumes that 1/2 LSB error is used (4,096 steps for the ADC). The 1/2 LSB error is the maximum error allowed for the ADC to meet its specified resolution.

### EQUATION 37-1: ACQUISITION TIME EXAMPLE

*Assumptions:* Temperature = 50°C and external impedance of 10kΩ 5.0V VDD

$$\begin{aligned} T_{ACQ} &= \text{Amplifier Settling Time} + \text{Hold Capacitor Charging Time} + \text{Temperature Coefficient} \\ &= T_{AMP} + T_C + T_{COFF} \\ &= 2\mu s + T_C + [( \text{Temperature} - 25^\circ\text{C} )(0.05\mu s/^\circ\text{C})] \end{aligned}$$

*The value for TC can be approximated with the following equations:*

$$V_{APPLIED} \left( 1 - \frac{1}{(2^{n+1}) - 1} \right) = V_{CHOLD} \quad ;[1] \text{ } V_{CHOLD} \text{ charged to within } 1/2 \text{ lsb}$$

$$V_{APPLIED} \left( 1 - e^{\frac{-T_C}{RC}} \right) = V_{CHOLD} \quad ;[2] \text{ } V_{CHOLD} \text{ charge response to } V_{APPLIED}$$

$$V_{APPLIED} \left( 1 - e^{\frac{-T_C}{RC}} \right) = V_{APPLIED} \left( 1 - \frac{1}{(2^{n+1}) - 1} \right) \quad ;\text{combining [1] and [2]}$$

*Note: Where n = number of bits of the ADC.*

*Solving for TC:*

$$\begin{aligned} T_C &= -CHOLD(R_{IC} + R_{SS} + R_S) \ln(1/8191) \\ &= -28\text{pF}(1\text{k}\Omega + 7\text{k}\Omega + 10\text{k}\Omega) \ln(0.0001221) \\ &= 4.54\mu s \end{aligned}$$

*Therefore:*

$$\begin{aligned} T_{ACQ} &= 2\mu s + 4.54\mu s + [(50^\circ\text{C} - 25^\circ\text{C})(0.05\mu s/^\circ\text{C})] \\ &= 7.79\mu s \end{aligned}$$

**Note 1:** The reference voltage (VREF) has no effect on the equation, since it cancels itself out.

**2:** The charge holding capacitor (CHOLD) is not discharged after each conversion.

**3:** The maximum recommended impedance for analog sources is 10 kΩ. This is required to meet the pin leakage specification.

**FIGURE 37-4: ANALOG INPUT MODEL**

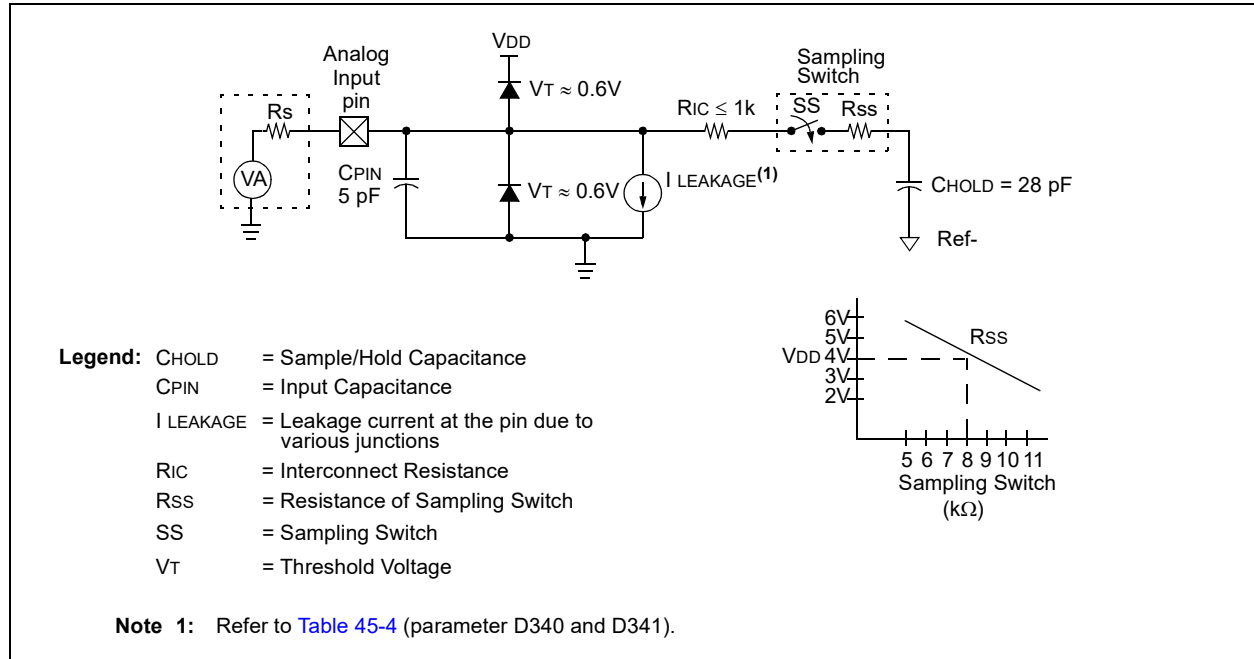
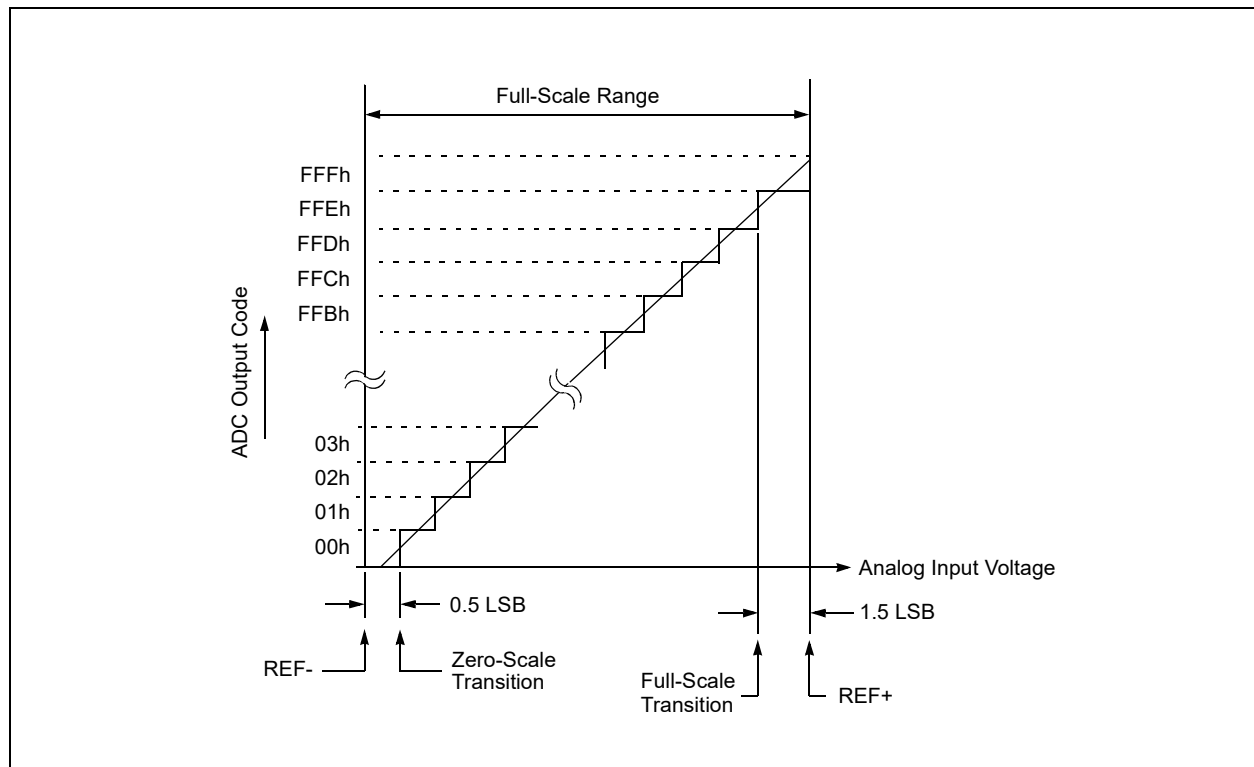


**FIGURE 37-5: ADC TRANSFER FUNCTION**



## 37.4 ADC Charge Pump

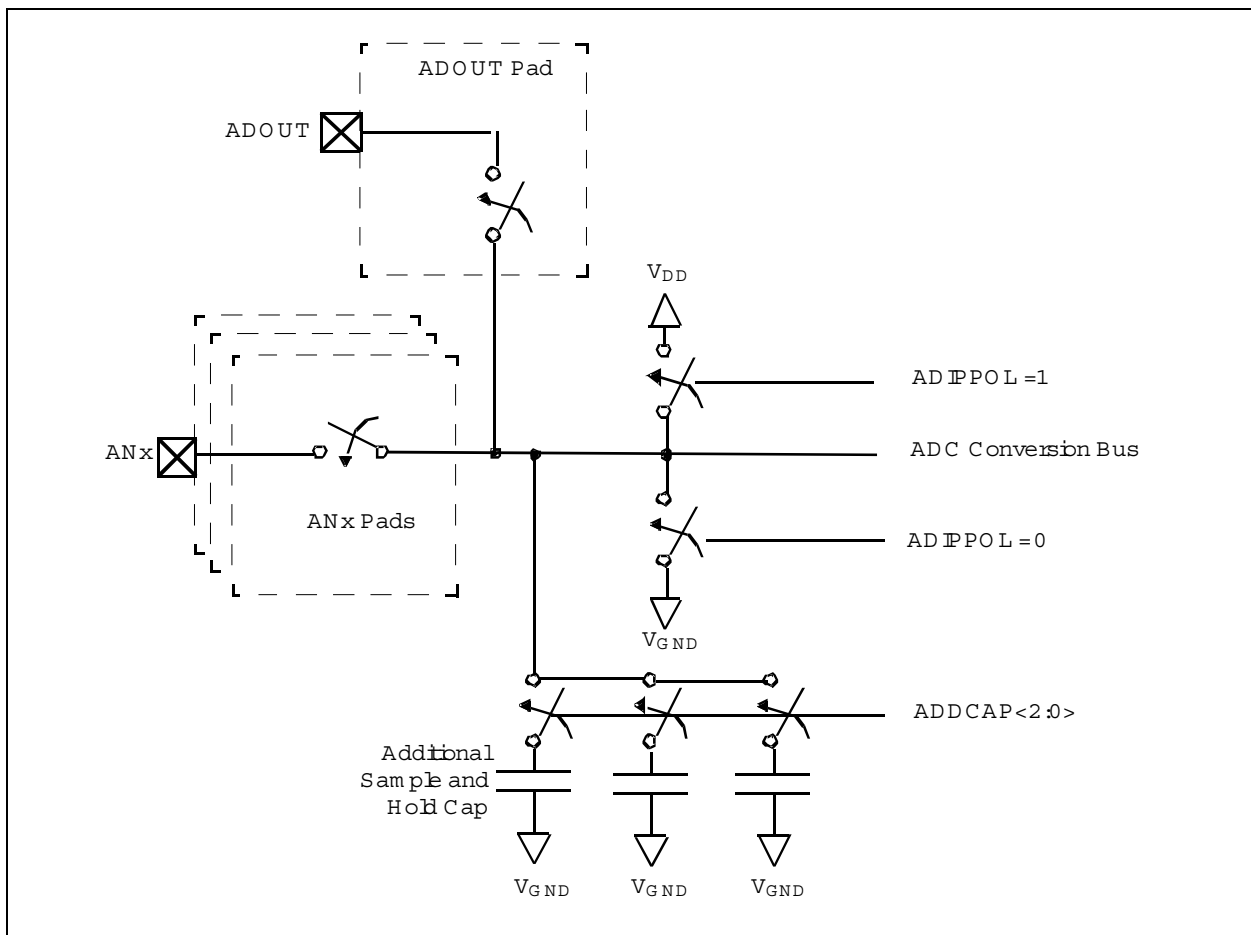
The ADC module has a dedicated charge pump which can be controlled through the ADCP register ([Register 37-36](#)). The primary purpose of the charge pump is to supply a constant voltage to the gates of transistor devices in the A/D converter, signal and reference input pass-gates, to prevent degradation of transistor performance at low operating voltage.

The charge pump can be enabled by setting the CPON bit in the ADC register. Once enabled, the pump will undergo a start-up time to stabilize the charge pump output. Once the output stabilizes and is ready for use, the CPRDY bit of the ADCP register will be set.

## 37.5 Capacitive Voltage Divider (CVD) Features

The ADC module contains several features that allow the user to perform a relative capacitance measurement on any ADC channel using the internal ADC sample and hold capacitance as a reference. This relative capacitance measurement can be used to implement capacitive touch or proximity sensing applications. [Figure 37-6](#) shows the basic block diagram of the CVD portion of the ADC module.

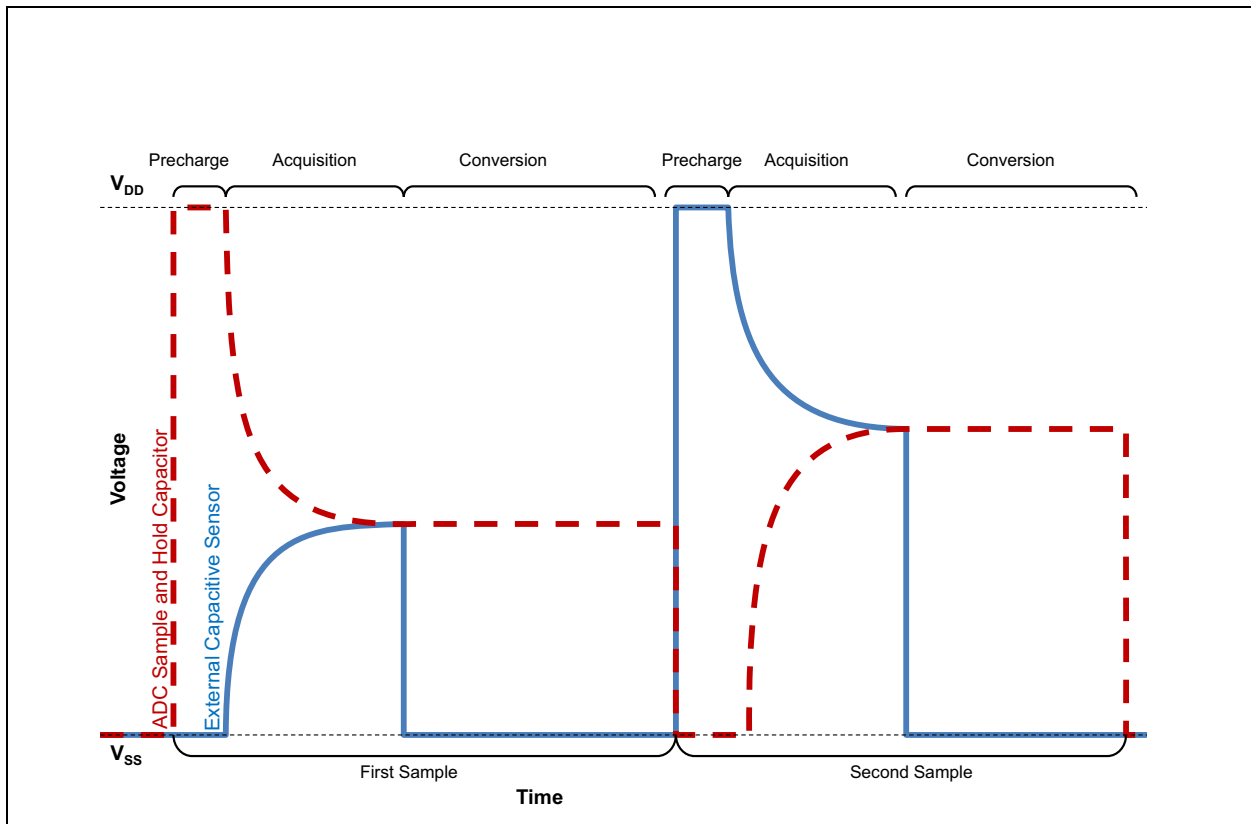
**FIGURE 37-6: HARDWARE CAPACITIVE VOLTAGE DIVIDER BLOCK DIAGRAM**



## 37.5.1 CVD OPERATION

A CVD operation begins with the ADC's internal sample and hold capacitor ( $C_{\text{HOLD}}$ ) being disconnected from the path which connects it to the external capacitive sensor node. While disconnected,  $C_{\text{HOLD}}$  is precharged to  $V_{\text{DD}}$  or  $V_{\text{SS}}$ , while the path to the sensor node is precharged to the level opposite that of  $C_{\text{HOLD}}$ . When the precharge phase is complete, the  $V_{\text{DD}}/V_{\text{SS}}$  precharge paths for the two nodes are shut off and  $C_{\text{HOLD}}$  and the path to the external sensor node are reconnected, at which time the acquisition phase of the CVD operation begins. During acquisition, a capacitive voltage divider is formed between the precharged  $C_{\text{HOLD}}$  and sensor nodes, which results in a final voltage level setting on  $C_{\text{HOLD}}$ , which is determined by the capacitances and precharge levels of the two nodes. After acquisition, the ADC converts the voltage level on  $C_{\text{HOLD}}$ . This process is then repeated with inverted precharge levels for both the  $C_{\text{HOLD}}$  and external sensor nodes. Figure 37-7 shows the waveform for two inverted CVD measurements, which is known as differential CVD measurement.

**FIGURE 37-7: DIFFERENTIAL CVD MEASUREMENT WAVEFORM**





## 37.5.2 PRECHARGE CONTROL

The precharge stage is an optional period of time that brings the external channel and internal sample and hold capacitor to known voltage levels. Precharge is enabled by writing a non-zero value to the PRE register. This stage is initiated when an ADC conversion begins, either from setting the GO bit, a special event trigger, or a conversion restart from the computation functionality. If the PRE register is cleared when an ADC conversion begins, this stage is skipped.

During the precharge time, CHOLD is disconnected from the outer portion of the sample path that leads to the external capacitive sensor and is connected to either VDD or VSS, depending on the value of the ADPPOL bit of ADCON1. At the same time, the port pin logic of the selected analog channel is overridden to drive a digital high or low out, in order to precharge the outer portion of the ADC's sample path, which includes the external sensor. The output polarity of this override is also determined by the ADPPOL bit of ADCON1. The amount of time that this charging receives is controlled by the PRE register.

- Note 1:** The external charging overrides the TRIS setting of the respective I/O pin.
- 2:** If there is a device attached to this pin, Precharge should not be used.

## 37.5.3 ACQUISITION CONTROL

The Acquisition stage is an optional time for the voltage on the internal sample and hold capacitor to charge or discharge from the selected analog channel. This acquisition time is controlled by the ADACQ register. If PRE = 0, acquisition starts at the beginning of conversion. When PRE = 1, the acquisition stage begins when precharge ends.

At the start of the acquisition stage, the port pin logic of the selected analog channel is overridden to turn off the digital high/low output drivers so they do not affect the final result of the charge averaging. Also, the selected ADC channel is connected to CHOLD. This allows charge averaging to proceed between the precharged channel and the CHOLD capacitor.

**Note:** When PRE! = 0, acquisition time cannot be '0'. In this case, setting ADACQ to '0' will set a maximum acquisition time (8191 ADC clock cycles). When precharge is disabled, setting ADACQ to '0' will disable hardware acquisition time control.

## 37.5.4 GUARD RING OUTPUTS

Figure 37-8 shows a typical guard ring circuit. CGUARD represents the capacitance of the guard ring trace placed on the PCB board. The user selects values for RA and RB that will create a voltage profile on CGUARD, which will match the selected acquisition channel.

The purpose of the guard ring is to generate a signal in phase with the CVD sensing signal to minimize the effects of the parasitic capacitance on sensing electrodes. It also can be used as a mutual drive for mutual capacitive sensing. For more information about active guard and mutual drive, see Application Note AN1478, "mTouch™ Sensing Solution Acquisition Methods Capacitive Voltage Divider" (DS01478).

The ADC has two guard ring drive outputs, ADGRDA and ADGRDB. These outputs can be routed through PPS controls to I/O pins (see Section 17.0 "Peripheral Pin Select (PPS) Module" for details) and the polarity of these outputs are controlled by the ADGPOL and ADIPEN bits of ADCON1.

At the start of the first precharge stage, both outputs are set to match the ADGPOL bit of ADCON1. Once the acquisition stage begins, ADGRDA changes polarity, while ADGRDB remains unchanged. When performing a double sample conversion, setting the ADIPEN bit of ADCON1 causes both guard ring outputs to transition to the opposite polarity of ADGPOL at the start of the second precharge stage, and ADGRDA toggles again for the second acquisition. For more information on the timing of the guard ring output, refer to Figure 37-8 and Figure 37-9.

**FIGURE 37-8: GUARD RING CIRCUIT**

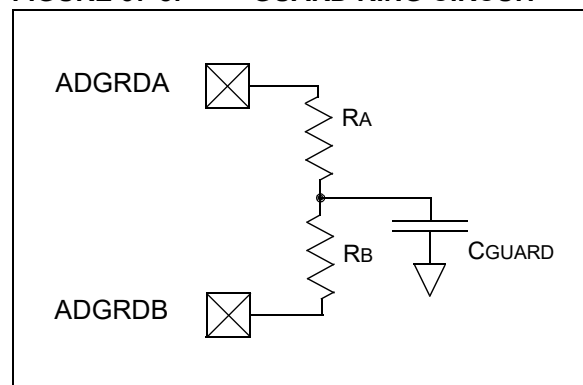
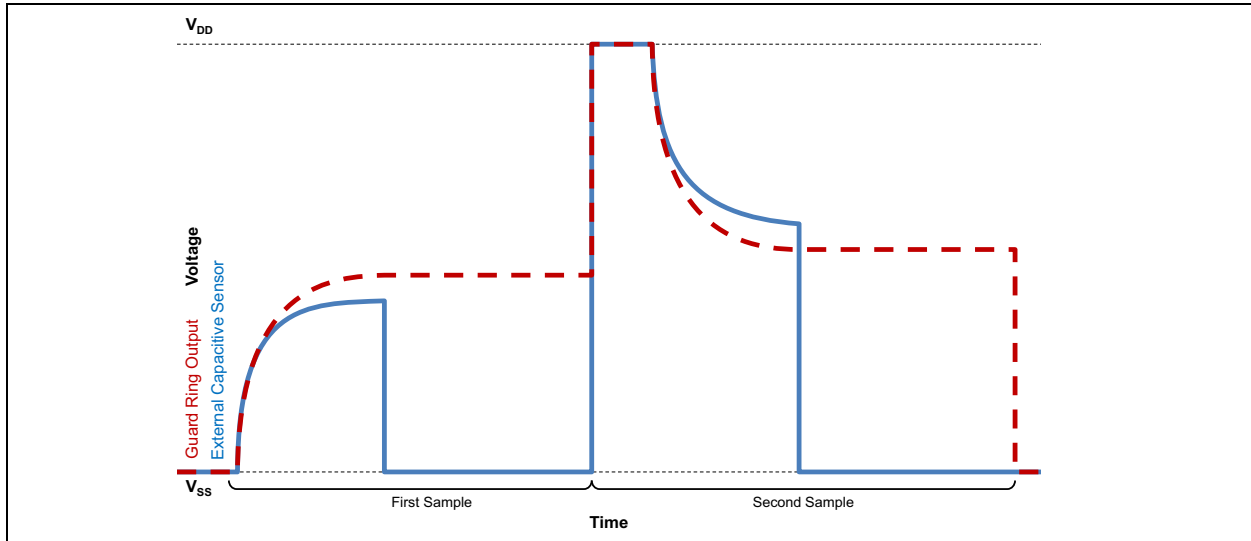


**FIGURE 37-9: DIFFERENTIAL CVD WITH GUARD RING OUTPUT WAVEFORM**



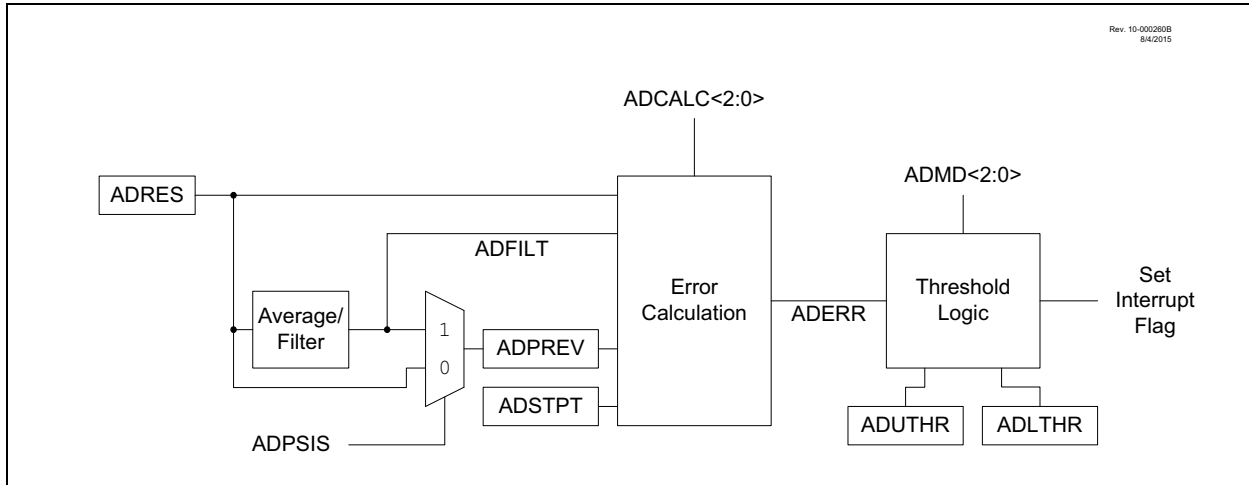
## 37.5.5 ADDITIONAL SAMPLE AND HOLD CAPACITANCE

Additional capacitance can be added in parallel with the internal sample and hold capacitor (CHOLD) by using the ADCAP register. This register selects a digitally programmable capacitance which is added to the ADC conversion bus, increasing the effective internal capacitance of the sample and hold capacitor in the ADC module. This is used to improve the match between internal and external capacitance for a better sensing performance. The additional capacitance does not affect analog performance of the ADC because it is not connected during conversion. See [Figure 37-10](#).

## 37.6 Computation Operation

The ADC module hardware is equipped with post conversion computation features. These features provide data post-processing functions that can be operated on the ADC conversion result, including digital filtering/averaging and threshold comparison functions.

**FIGURE 37-10: COMPUTATIONAL FEATURES SIMPLIFIED BLOCK DIAGRAM**



The operation of the ADC computational features is controlled by ADM <2:0> bits in the ADCON2 register.

The module can be operated in one of five modes:

- **Basic:** In this mode, ADC conversion occurs on single (ADSEN = 0) or double (ADSEN = 1) samples. ADIF is set after all the conversion are complete.
- **Accumulate:** With each trigger, the ADC conversion result is added to accumulator and CNT increments. ADIF is set after each conversion. ADTIF is set according to the calculation mode.
- **Average:** With each trigger, the ADC conversion result is added to the accumulator. When the RPT number of samples have been accumulated, a threshold test is performed. Upon the next trigger, the accumulator is cleared. For the subsequent tests, additional RPT samples are required to be accumulated.
- **Burst Average:** At the trigger, the accumulator is cleared. The ADC conversion results are then collected repetitively until RPT samples are accumulated and finally the threshold is tested.
- **Low-Pass Filter (LPF):** With each trigger, the ADC conversion result is sent through a filter. When RPT samples have occurred, a threshold test is performed. Every trigger after that the ADC conversion result is sent through the filter and another threshold test is performed.

The five modes are summarized in [Table 37-2](#) below.

**TABLE 37-2: COMPUTATION MODES**

| Mode            | ADMD | Bit Clear Conditions                      | Value after Trigger completion                                           |                                                      | Threshold Operations |                |                   |       |
|-----------------|------|-------------------------------------------|--------------------------------------------------------------------------|------------------------------------------------------|----------------------|----------------|-------------------|-------|
|                 |      |                                           | ACC                                                                      | CNT                                                  | Retrigger            | Threshold Test | Interrupt         | ADRES |
| Basic           | 0    | ADACLR = 1                                | Unchanged                                                                | Unchanged                                            | No                   | Every Sample   | If threshold=true | ACC   |
| Accumulate      | 1    | ADACLR = 1                                | S + ACC<br>or<br>(S2-S1) + ACC                                           | If (CNT=0xFF): CNT, otherwise: CNT+1                 | No                   | Every Sample   | If threshold=true | ACC   |
| Average         | 2    | ADACLR = 1 or CNT>=RPT at GO or retrigger | S + ACC<br>or<br>(S2-S1) + ACC                                           | If (CNT=0xFF): CNT, otherwise: CNT+1                 | No                   | If CNT>=RPT    | If threshold=true | ACC   |
| Burst Average   | 3    | ADACLR = 1 or GO set or retrigger         | Each repetition: same as Average<br>End with sum of all samples          | Each repetition: same as Average<br>End with CNT=RPT | Repeat while CNT<RPT | If CNT>=RPT    | If threshold=true | ACC   |
| Low-pass Filter | 4    | ADACLR = 1                                | S+ACC-ACC/2 <sup>ADCRS</sup><br>or<br>(S2-S1)+ACC-ACC/2 <sup>ADCRS</sup> | Count up, stop counting when CNT = 0xFF              | No                   | If CNT>=RPT    | If threshold=true | ACC   |

**Note:** S1 and S2 are abbreviations for Sample 1 and Sample 2, respectively. When ADDSEN = 0, S1 = ADRES; When ADDSEN = 1, S1 = PREV and S2 = ADRES.

## 37.6.1 DIGITAL FILTER/AVERAGE

The digital filter/average module consists of an accumulator with data feedback options, and control logic to determine when threshold tests need to be applied. The accumulator is a 16-bit wide register which can be accessed through the ADACCH:ADACCL register pair.

Upon each trigger event (the GO bit set or external event trigger), the ADC conversion result is added to the accumulator. If the accumulated result exceeds  $2^{(\text{accumulator\_width}-1)} = 18 = 262143$ , the overflow bit ADAOV in the ADSTAT register is set.

The number of samples to be accumulated is determined by the RPT (A/D Repeat Setting) register. Each time a sample is added to the accumulator, the ADCNT register is incremented. Once RPT samples are accumulated (CNT = RPT), an Accumulator Clear command can be issued by the software by setting the ADACLR bit in the ADCON2 register. Setting the ADACLR bit will also clear the ADAOV (Accumulator overflow) bit in the ADSTAT register, as well as the

ADCNT register. The ADACLR bit is cleared by the hardware when accumulator clearing action is complete.

**Note:** When ADC is operating from FRC, five FRC clock cycles are required to execute the ACC clearing operation.

The ADCRS <2:0> bits in the ADCON2 register control the data shift on the accumulator result, which effectively divides the value in accumulator (ADACCU:ADACCH:ADACCL) register pair. For the Accumulate mode of the digital filter, the shift provides a simple scaling operation. For the Average/Burst Average mode, the shift bits are used to determine the number of logical right shifts to be performed on the accumulated result. For the Low-pass Filter mode, the shift is an integral part of the filter, and determines the cut-off frequency of the filter. Table 37-3 shows the -3 dB cut-off frequency in  $\omega T$  (radians) and the highest signal attenuation obtained by this filter at nyquist frequency ( $\omega T = \pi$ ).

**TABLE 37-3: LOW-PASS FILTER -3 dB CUT-OFF FREQUENCY**

| ADCRS | $\omega T$ (radians) @ -3 dB Frequency | dB @ $F_{\text{nyquist}}=1/(2T)$ |
|-------|----------------------------------------|----------------------------------|
| 1     | 0.72                                   | -9.5                             |
| 2     | 0.284                                  | -16.9                            |
| 3     | 0.134                                  | -23.5                            |
| 4     | 0.065                                  | -29.8                            |
| 5     | 0.032                                  | -36.0                            |
| 6     | 0.016                                  | -42.0                            |
| 7     | 0.0078                                 | -48.1                            |

## 37.6.2 BASIC MODE

Basic mode (ADMD = 000) disables all additional computation features. In this mode, no accumulation occurs but threshold error comparison is performed. Double sampling, Continuous mode, and all CVD features are still available, but no features involving the digital filter/average features are used.

## 37.6.3 ACCUMULATE MODE

In Accumulate mode (ADMD = 001), after every conversion, the ADC result is added to the ADACC register. The ADACC register is right-shifted by the value of the ADCRS bits in the ADCON2 register. This right-shifted value is copied in to the ADFLT register. The Formatting mode does not affect the right-justification of the ACC value. Upon each sample, CNT is also incremented, incrementing the number of samples accumulated. After each sample and accumulation, the ACC value has a threshold comparison performed on it (see Section 37.6.7 “Threshold Comparison”) and the ADTIF interrupt may trigger.

## 37.6.4 AVERAGE MODE

In Average mode (ADMD = 010), the ADACC registers accumulate with each ADC sample, much as in Accumulate mode, and the ADCNT register increments with each sample. The ADFLT register is also updated with the right-shifted value of the ADACC register. The value of the ADCRS bits governs the number of right shifts. However, in Average mode, the threshold comparison is performed upon CNT being greater than or equal to a user-defined RPT value. In this mode when  $RPT = 2^{\text{CNT}}$ , then the final accumulated value will be divided by number of samples, allowing for a threshold comparison operation on the average of all gathered samples.

## 37.6.5 BURST AVERAGE MODE

The Burst Average mode (ADMD = 011) acts the same as the Average mode in most respects. The one way it differs is that it continuously retriggers ADC sampling until the CNT value is greater than or equal to RPT, even if Continuous Sampling mode (see [Section 37.6.8 “Continuous Sampling mode”](#)) is not enabled. This allows for a threshold comparison on the average of a short burst of ADC samples.

## 37.6.6 LOW-PASS FILTER MODE

The Low-pass Filter mode (ADMD = 100) acts similarly to the Average mode in how it handles samples (accumulates samples until CNT value greater than or equal to RPT, then triggers threshold comparison), but instead of a simple average, it performs a low-pass filter operation on all of the samples, reducing the effect of high-frequency noise on the average, then performs a threshold comparison on the results. (see [Table 37-2](#) for a more detailed description of the mathematical operation). In this mode, the ADCRS bits determine the cut-off frequency of the low-pass filter (as demonstrated by [Table 37-3](#)).

## 37.6.7 THRESHOLD COMPARISON

At the end of each computation:

- The conversion results are latched and held stable at the end-of-conversion.
- The error is calculated based on a difference calculation which is selected by the ADCALC<2:0> bits in the ADCON3 register. The value can be one of the following calculations (see [Register 37-4](#) for more details):
  - The first derivative of single measurements
  - The CVD result in CVD mode
  - The current result vs. a setpoint
  - The current result vs. the filtered/average result
  - The first derivative of the filtered/average value
  - Filtered/average value vs. a setpoint
- The result of the calculation (ERR) is compared to the upper and lower thresholds, UTH<ADUTHH:ADUTHL> and LTH<ADLTHH:ADLTHL> registers, to set the ADUTHR and ADLTHR flag bits. The threshold logic is selected by ADTMD<2:0> bits in the ADCON3 register. The threshold trigger option can be one of the following:
  - Never interrupt
  - Error is less than lower threshold
  - Error is greater than or equal to lower threshold
  - Error is between thresholds (inclusive)
  - Error is outside of thresholds
  - Error is less than or equal to upper threshold
  - Error is greater than upper threshold

- Always interrupt regardless of threshold test results
- If the threshold condition is met, the threshold interrupt flag ADTIF is set.

**Note 1:** The threshold tests are signed operations.

**2:** If ADAOV is set, a threshold interrupt is signaled.

## 37.6.8 CONTINUOUS SAMPLING MODE

Setting the CONT bit in the ADCON0 register automatically retriggers a new conversion cycle after updating the ADACC register. The GO bit remains set and retriggering occurs automatically.

If ADSOI = 1, a threshold interrupt condition will clear GO and the conversions will stop.

## 37.6.9 DOUBLE SAMPLE CONVERSION

Double sampling is enabled by setting the ADDSEN bit of the ADCON1 register. When this bit is set, two conversions are required before the module will calculate threshold error (each conversion must still be triggered separately). The first conversion will set the ADMATH bit of the ADSTAT register and update ADACC, but will not calculate ERR or trigger ADTIF. When the second conversion completes, the first value is transferred to PREV (depending on the setting of ADPSIS) and the value of the second conversion is placed into ADRES. Only upon the completion of the second conversion is ERR calculated and ADTIF triggered (depending on the value of ADCALC).

## 37.7 Register Definitions: ADC Control

**REGISTER 37-1: ADCON0: ADC CONTROL REGISTER 0**

| R/W-0/0 | R/W-0/0 | U-0 | R/W-0/0 | U-0 | R/W-0/0 | U-0 | R/W/HC-0 |
|---------|---------|-----|---------|-----|---------|-----|----------|
| ON      | CONT    | —   | CS      | —   | FM      | —   | GO       |
| bit 7   |         |     |         |     |         |     | bit 0    |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

HC = Bit is cleared by hardware

bit 7 **ON:** ADC Enable bit

1 = ADC is enabled

0 = ADC is disabled

bit 6 **CONT:** ADC Continuous Operation Enable bit

1 = GO is retriggered upon completion of each conversion trigger until ADTIF is set (if ADSOI is set) or until GO is cleared (regardless of the value of ADSOI)

0 = ADC is cleared upon completion of each conversion trigger

bit 5 **Unimplemented:** Read as '0'

bit 4 **CS:** ADC Clock Selection bit

1 = Clock supplied from FRC dedicated oscillator

0 = Clock supplied by FOSC, divided according to ADCLK register

bit 3 **Unimplemented:** Read as '0'

bit 2 **FM:** ADC results Format/alignment Selection

1 = ADRES and PREV data are right-justified

0 = ADRES and PREV data are left-justified, zero-filled

bit 1 **Unimplemented:** Read as '0'

bit 0 **GO:** ADC Conversion Status bit<sup>(1)</sup>

1 = ADC conversion cycle in progress. Setting this bit starts an ADC conversion cycle. The bit is cleared by hardware as determined by the CONT bit

0 = ADC conversion completed/not in progress

**Note 1:** This bit requires ON bit to be set.

**2:** If cleared by software while a conversion is in progress, the results of the conversion up to this point will be transferred to ADRES and the state machine will be reset, but the ADIF interrupt flag bit will not be set; filter and threshold operations will not be performed.

## REGISTER 37-2: ADCON1: ADC CONTROL REGISTER 1

|         |         |         |     |     |     |     |         |
|---------|---------|---------|-----|-----|-----|-----|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | U-0 | U-0 | U-0 | U-0 | R/W-0/0 |
| PPOL    | IPEN    | GPOL    | —   | —   | —   | —   | DSEN    |
| bit 7   |         |         |     |     |     |     | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7

**PPOL:** Precharge Polarity bit

If PRE>0x00:

| PPOL | Action During 1st Precharge Stage  |                                    |
|------|------------------------------------|------------------------------------|
|      | External (selected analog I/O pin) | Internal (AD sampling capacitor)   |
| 1    | Connected to VDD                   | C <sub>HOLD</sub> connected to VSS |
| 0    | Connected to VSS                   | C <sub>HOLD</sub> connected to VDD |

Otherwise:

The bit is ignored

bit 6

**IPEN:** A/D Inverted Precharge Enable bit

If DSEN = 1

1 = The precharge and guard signals in the second conversion cycle are the opposite polarity of the first cycle

0 = Both Conversion cycles use the precharge and guards specified by ADPPOL and ADGPOL

Otherwise:

The bit is ignored

bit 5

**GPOL:** Guard Ring Polarity Selection bit

1 = ADC guard Ring outputs start as digital high during Precharge stage

0 = ADC guard Ring outputs start as digital low during Precharge stage

bit 4-1

**Unimplemented:** Read as '0'

bit 0

**DSEN:** Double-sample enable bit

1 = Two conversions are performed on each trigger. Data from the first conversion appears in PREV

0 = One conversion is performed for each trigger



## REGISTER 37-3: ADCON2: ADC CONTROL REGISTER 2

|         |          |         |         |          |         |         |         |
|---------|----------|---------|---------|----------|---------|---------|---------|
| R/W-0/0 | R/W-0/0  | R/W-0/0 | R/W-0/0 | R/W/HC-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| PSIS    | CRS<2:0> |         |         | ACLR     | MD<2:0> |         |         |
| bit 7   |          |         |         |          |         |         | bit 0   |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared | HC = Bit is cleared by hardware                       |

- bit 7      **PSIS:** ADC Previous Sample Input Select bits  
1 = PREV is the FLTR value at start-of-conversion  
0 = PREV is the RES value at start-of-conversion
- bit 6-4      **CRS<2:0>:** ADC Accumulated Calculation Right Shift Select bits  
If ADMD = 100:  
Low-pass filter time constant is  $2^{\text{ADCRS}}$ , filter gain is 1:1  
If ADMD = 001, 010 or 011:  
The accumulated value is right-shifted by CRS (divided by  $2^{\text{ADCRS}}$ )<sup>(1,2)</sup>  
Otherwise:  
Bits are ignored
- bit 3      **ACLR:** A/D Accumulator Clear Command bit<sup>(3)</sup>  
1 = ACC, AOV and CNT registers are cleared  
0 = Clearing action is complete (or not started)
- bit 2-0      **MD<2:0>:** ADC Operating Mode Selection bits<sup>(4)</sup>  
111-101 = Reserved  
100 = Low-pass Filter mode  
011 = Burst Average mode  
010 = Average mode  
001 = Accumulate mode  
000 = Basic mode

- Note 1:** To correctly calculate an average, the number of samples (set in RPT) must be  $2^{\text{ADCRS}}$ .
- 2:** ADCRS = 3 'b111 is a reserved option.
- 3:** This bit is cleared by hardware when the accumulator operation is complete; depending on oscillator selections, the delay may be many instructions.
- 4:** See [Table 37-2](#) for Full mode descriptions.

## REGISTER 37-4: ADCON3: ADC CONTROL REGISTER 3

|       |           |         |         |          |          |         |         |
|-------|-----------|---------|---------|----------|----------|---------|---------|
| U-0   | R/W-0/0   | R/W-0/0 | R/W-0/0 | R/W/HC-0 | R/W-0/0  | R/W-0/0 | R/W-0/0 |
| —     | CALC<2:0> |         |         | SOI      | TMD<2:0> |         |         |
| bit 7 |           |         |         |          |          |         | bit 0   |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared | HC = Bit is cleared by hardware                       |

bit 7 **Unimplemented:** Read as '0'

bit 6-4 **CALC<2:0>:** ADC Error Calculation Mode Select bits

| CALC | DSEN = 0<br>Single-Sample Mode | DSEN = 1 CVD<br>Double-Sample Mode <sup>(1)</sup> | Application                                                  |
|------|--------------------------------|---------------------------------------------------|--------------------------------------------------------------|
| 111  | Reserved                       | Reserved                                          | Reserved                                                     |
| 110  | Reserved                       | Reserved                                          | Reserved                                                     |
| 101  | FLTR-STPT                      | FLTR-STPT                                         | Average/filtered value vs. setpoint                          |
| 100  | PREV-FLTR                      | PREV-FLTR                                         | First derivative of filtered value <sup>(3)</sup> (negative) |
| 011  | Reserved                       | Reserved                                          | Reserved                                                     |
| 010  | RES-FLTR                       | (RES-PREV)-FLTR                                   | Actual result vs. averaged/filtered value                    |
| 001  | RES-STPT                       | (RES-PREV)-STPT                                   | Actual result vs. setpoint                                   |
| 000  | RES-PREV                       | RES-PREV                                          | First derivative of single measurement <sup>(2)</sup>        |
|      |                                |                                                   | Actual CVD result in CVD mode <sup>(2)</sup>                 |

bit 3 **SOI:** ADC Stop-on-Interrupt bit

If **CONT = 1**:

1 = GO is cleared when the threshold conditions are met, otherwise the conversion is retrigged  
0 = GO is not cleared by hardware, must be cleared by software to stop retriggers

bit 2-0 **TMD<2:0>:** Threshold Interrupt Mode Select bits

111 = Interrupt regardless of threshold test results  
110 = Interrupt if ERR>UTH  
101 = Interrupt if ERR≤UTH  
100 = Interrupt if ERR<LTH or ERR>UTH  
011 = Interrupt if ERR>LTH and ERR<UTH  
010 = Interrupt if ERR≥LTH  
001 = Interrupt if ERR<LTH  
000 = Never interrupt

**Note 1:** When PSIS = 0, the value of (RES-PREV) is the value of (S2-S1) from [Table 37-2](#).

**2:** When PSIS = 0

**3:** When PSIS = 1.

## REGISTER 37-5: ADSTAT: ADC STATUS REGISTER

|       |       |       |             |     |           |       |       |
|-------|-------|-------|-------------|-----|-----------|-------|-------|
| R-0/0 | R-0/0 | R-0/0 | R/HS/HC-0/0 | U-0 | R-0/0     | R-0/0 | R-0/0 |
| AOV   | UTHR  | LTHR  | MATH        | —   | STAT<2:0> |       |       |
| bit 7 |       |       |             |     |           |       | bit 0 |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared | HS/HC = Bit is set/cleared by hardware                |

|         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| bit 7   | <b>AOV:</b> ADC Accumulator Overflow bit<br>1 = ADC accumulator or ERR calculation have overflowed<br>0 = ADC accumulator and ERR calculation have not overflowed                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| bit 6   | <b>UTHR:</b> ADC Module Greater-than Upper Threshold Flag bit<br>1 = ERR > UTH<br>0 = ERR ≤ UTH                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| bit 5   | <b>LTHR:</b> ADC Module Less-than Lower Threshold Flag bit<br>1 = ERR < LTH<br>0 = ERR ≥ LTH                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| bit 4   | <b>MATH:</b> ADC Module Computation Status bit<br>1 = Registers ACC, FLTR, UTH, LTH and the AOV bit are updating or have already updated<br>0 = Associated registers/bits have not changed since this bit was last cleared                                                                                                                                                                                                                                                                                                                                                                                                                             |
| bit 3   | <b>Unimplemented:</b> Read as '0'                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| bit 2-0 | <b>STAT&lt;2:0&gt;:</b> ADC Module Cycle Multistage Status bits <sup>(1)</sup><br>111 = ADC module is in 2 <sup>nd</sup> conversion stage<br>110 = ADC module is in 2 <sup>nd</sup> acquisition stage<br>101 = ADC module is in 2 <sup>nd</sup> precharge stage<br>100 = ADC computation is suspended between 1st and 2nd sample: the computation results are incomplete and awaiting data from the 2nd sample <sup>(2,3)</sup><br>011 = ADC module is in 1 <sup>st</sup> conversion stage<br>010 = ADC module is in 1 <sup>st</sup> acquisition stage<br>001 = ADC module is in 1 <sup>st</sup> precharge stage<br>000 = ADC module is not converting |

**Note 1:** If CS = 1, and FOSC < FRC, these bits may be invalid.

**2:** If ADOSC = ADCRC and FOSC < FRC, this reading may be invalid.

**3:** ADSTAT = 100 appears between the two triggers when ADDSEN = 1 and ADCONT = 0.

## REGISTER 37-6: ADCLK: ADC CLOCK SELECTION REGISTER

| U-0   | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|-------|-----|---------|---------|---------|---------|---------|---------|
| —     | —   | CS<5:0> |         |         |         |         |         |
| bit 7 |     |         |         |         |         |         | bit 0   |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-6 **Unimplemented:** Read as '0'

bit 5-0 **CS<5:0>:** ADC Conversion Clock Select bits

111111 = Fosc/128

111110 = Fosc/126

111101 = Fosc/124

•

•

•

000000 = Fosc/2

## REGISTER 37-7: ADREF: ADC REFERENCE SELECTION REGISTER

| U-0   | U-0 | U-0 | R/W-0/0 | U-0 | U-0 | R/W-0/0   | R/W-0/0 |
|-------|-----|-----|---------|-----|-----|-----------|---------|
| —     | —   | —   | NREF    | —   | —   | PREF<1:0> |         |
| bit 7 |     |     |         |     |     |           | bit 0   |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-5 **Unimplemented:** Read as '0'

bit 4 **NREF:** ADC Negative Voltage Reference Selection bit

1 = VREF- is connected to external VREF-

0 = VREF- is connected to VSS

bit 3-2 **Unimplemented:** Read as '0'

bit 1-0 **PREF:** ADC Positive Voltage Reference Selection bits

11 = VREF+ is connected to internal Fixed Voltage Reference (FVR) module

10 = VREF+ is connected to external VREF+

01 = Reserved

00 = VREF+ is connected to VDD

## REGISTER 37-8: ADPCH: ADC POSITIVE CHANNEL SELECTION REGISTER

| U-0   | U-0 | R/W-0/0    | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|-------|-----|------------|---------|---------|---------|---------|---------|
| —     | —   | ADPCH<5:0> |         |         |         |         |         |
| bit 7 |     |            |         |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-6 **Unimplemented:** Read as '0'

bit 5-0 **ADPCH<5:0>:** ADC Positive Input Channel Selection bits

111111 = Fixed Voltage Reference (FVR)<sup>(2)</sup>

111110 = DAC1 output<sup>(1)</sup>

111101 = Temperature Indicator<sup>(3)</sup>

111100 = AVss (Analog Ground)

111011 = Reserved. No channel connected.

•  
•  
•

010111 = ANC7

010110 = ANC6

010101 = ANC5

010100 = ANC4

010011 = ANC3

010010 = ANC2

010001 = ANC1

010000 = ANC0

001111 = ANB7

001110 = ANB6

001101 = ANB5

001100 = ANB4

001011 = ANB3

001010 = ANB2

001001 = ANB1

001000 = ANB0

000111 = ANA7

000110 = ANA6

000101 = ANA5

000100 = ANA4

000011 = ANA3

000010 = ANA2

000001 = ANA1

000000 = ANA0

**Note 1:** See [Section 38.0 “5-Bit Digital-to-Analog Converter \(DAC\) Module”](#) for more information.

**Note 2:** See [Section 35.0 “Fixed Voltage Reference \(FVR\)”](#) for more information.

**Note 3:** See [Section 36.0 “Temperature Indicator Module”](#) for more information.

**REGISTER 37-9: ADPREL: ADC PRECHARGE TIME CONTROL REGISTER (LOW BYTE)**

|          |         |         |         |         |         |         |         |
|----------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0  | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| PRE<7:0> |         |         |         |         |         |         |         |
| bit 7    |         |         |         | bit 0   |         |         |         |

**Legend:**

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0      **PRE<7:0>**: Precharge Time Select bits  
See [Table 37-4](#).

**REGISTER 37-10: ADPREH: ADC PRECHARGE TIME CONTROL REGISTER (HIGH BYTE)**

|       |     |     |           |         |         |         |         |
|-------|-----|-----|-----------|---------|---------|---------|---------|
| U-0   | U-0 | U-0 | R/W-0/0   | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| —     | —   | —   | PRE<12:8> |         |         |         |         |
| bit 7 |     |     |           | bit 0   |         |         |         |

**Legend:**

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-5      **Unimplemented**: Read as '0'  
bit 4-0      **PRE<12:8>**: Precharge Time Select bits  
See [Table 37-4](#).

**Note:** If PRE is not equal to '0', then ADACQ = b'00000000 means Acquisition time is 256 clocks of the selected ADC clock.

**TABLE 37-4: PRECHARGE TIME**

| ADPRE            | Precharge time                            |
|------------------|-------------------------------------------|
| 1 1111 1111 1111 | 8191 clocks of the selected ADC clock     |
| 1 1111 1111 1110 | 8190 clocks of the selected ADC clock     |
| 1 1111 1111 1101 | 8189 clocks of the selected ADC clock     |
| ...              | ...                                       |
| 0 0000 0000 0010 | 2 clocks of the selected ADC clock        |
| 0 0000 0000 0001 | 1 clock of the selected ADC clock         |
| 0 0000 0000 0000 | Not included in the data conversion cycle |

## REGISTER 37-11: ADACQL: ADC ACQUISITION TIME CONTROL REGISTER (LOW BYTE)

|          |         |         |         |         |         |         |         |
|----------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0  | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| ACQ<7:0> |         |         |         |         |         |         |         |
| bit 7    |         |         |         | bit 0   |         |         |         |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0 ACQ<7:0>: Acquisition (charge share time) Select bits  
See [Table](#) .

## REGISTER 37-12: ADACQH: ADC ACQUISITION TIME CONTROL REGISTER (HIGH BYTE)

|       |     |     |           |         |         |         |         |
|-------|-----|-----|-----------|---------|---------|---------|---------|
| U-0   | U-0 | U-0 | R/W-0/0   | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| —     | —   | —   | ACQ<12:8> |         |         |         |         |
| bit 7 |     |     |           | bit 0   |         |         |         |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-5 **Unimplemented:** Read as '0'  
bit 4-0 ACQ<12:8>: Acquisition (charge share time) Select bits  
See [Table](#) .

**TABLE 37-5: ACQUISITION TIME**

| ADACQ            | Acquisition time                                         |
|------------------|----------------------------------------------------------|
| 1 1111 1111 1111 | 8191 clocks of the selected ADC clock                    |
| 1 1111 1111 1110 | 8190 clocks of the selected ADC clock                    |
| 1 1111 1111 1101 | 8189 clocks of the selected ADC clock                    |
| ...              | ...                                                      |
| 0 0000 0000 0010 | 2 clocks of the selected ADC clock                       |
| 0 0000 0000 0001 | 1 clock of the selected ADC clock                        |
| 0 0000 0000 0000 | Not included in the data conversion cycle <sup>(1)</sup> |

**Note 1:** If ADPRE is not equal to '0', then ADACQ = 0b0\_0000\_0000\_0000 means Acquisition time is 8192 clocks of the selected ADC clock.

## REGISTER 37-13: ADCAP: ADC ADDITIONAL SAMPLE CAPACITOR SELECTION REGISTER

|       |     |     |            |         |         |         |         |
|-------|-----|-----|------------|---------|---------|---------|---------|
| U-0   | U-0 | U-0 | R/W-0/0    | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| —     | —   | —   | ADCAP<4:0> |         |         |         |         |
| bit 7 |     |     | bit 0      |         |         |         |         |



## REGISTER 37-15: ADCNT: ADC REPEAT COUNTER REGISTER

|          |         |         |         |         |         |         |         |
|----------|---------|---------|---------|---------|---------|---------|---------|
| R/W-x/u  | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
| CNT<7:0> |         |         |         |         |         |         |         |
| bit 7    |         |         |         | bit 0   |         |         |         |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0

**CNT<7:0>**: ADC Repeat Count bits

Determines the number of times that the ADC is triggered before the threshold is checked when the computation is Low-pass Filter, Burst Average, or Average modes. See [Table 37-2](#) for more details.

## REGISTER 37-16: ADFLTRH: ADC FILTER HIGH BYTE REGISTER

|            |     |     |     |       |     |     |     |
|------------|-----|-----|-----|-------|-----|-----|-----|
| R-x        | R-x | R-x | R-x | R-x   | R-x | R-x | R-x |
| FLTR<15:8> |     |     |     |       |     |     |     |
| bit 7      |     |     |     | bit 0 |     |     |     |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0

**FLTR<15:8>**: ADC Filter Output Most Significant bits

In Accumulate, Average, and Burst Average mode, this is equal to ACC right shifted by the ADCRS bits of ADCON2. In LPF mode, this is the output of the Low-pass Filter.

## REGISTER 37-17: ADFLTRL: ADC FILTER LOW BYTE REGISTER

|           |     |     |     |       |     |     |     |
|-----------|-----|-----|-----|-------|-----|-----|-----|
| R-x       | R-x | R-x | R-x | R-x   | R-x | R-x | R-x |
| FLTR<7:0> |     |     |     |       |     |     |     |
| bit 7     |     |     |     | bit 0 |     |     |     |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0

**FLTR<7:0>**: ADC Filter Output Least Significant bits

In Accumulate, Average, and Burst Average mode, this is equal to ACC right shifted by the ADCRS bits of ADCON2. In LPF mode, this is the output of the Low-pass Filter.

## REGISTER 37-18: ADRESH: ADC RESULT REGISTER HIGH, FM = 0

|             |         |         |         |         |         |         |         |
|-------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-x/u     | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
| ADRES<11:4> |         |         |         |         |         |         |         |
| bit 7       |         |         |         | bit 0   |         |         |         |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0      **ADRES<11:4>**: ADC Result Register bits  
Upper eight bits of 12-bit conversion result.

## REGISTER 37-19: ADRESL: ADC RESULT REGISTER LOW, FM = 0

|            |         |         |         |       |     |     |     |
|------------|---------|---------|---------|-------|-----|-----|-----|
| R/W-x/u    | R/W-x/u | R/W-x/u | R/W-x/u | U-0   | U-0 | U-0 | U-0 |
| ADRES<3:0> |         |         |         | —     | —   | —   | —   |
| bit 7      |         |         |         | bit 0 |     |     |     |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-4      **ADRES<3:0>**: ADC Result Register bits. Lower four bits of 12-bit conversion result.  
bit 3-0      **Reserved**

## REGISTER 37-20: ADRESH: ADC RESULT REGISTER HIGH, FM = 1

|       |     |     |     |             |         |         |         |
|-------|-----|-----|-----|-------------|---------|---------|---------|
| U-0   | U-0 | U-0 | U-0 | R/W-x/u     | R/W-x/u | R/W-x/u | R/W-x/u |
| —     | —   | —   | —   | ADRES<11:8> |         |         |         |
| bit 7 |     |     |     | bit 0       |         |         |         |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-4

**Reserved**

bit 3-0

**ADRES<11:8>**: ADC Sample Result bits. Upper four bits of 12-bit conversion result.

## REGISTER 37-21: ADRESL: ADC RESULT REGISTER LOW, FM = 1

|            |         |         |         |         |         |         |         |
|------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-x/u    | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
| ADRES<7:0> |         |         |         |         |         |         |         |
| bit 7      |         |         |         | bit 0   |         |         |         |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-0

**ADRES<7:0>**: ADC Result Register bits. Lower eight bits of 12-bit conversion result.

## REGISTER 37-22: ADPREVH: ADC PREVIOUS RESULT REGISTER

|                                                |     |     |     |     |     |     |     |
|------------------------------------------------|-----|-----|-----|-----|-----|-----|-----|
| R-x                                            | R-x | R-x | R-x | R-x | R-x | R-x | R-x |
| PREV<15:8>                                     |     |     |     |     |     |     |     |
| bit 7 <span style="float: right;">bit 0</span> |     |     |     |     |     |     |     |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0 **PREV<15:8>**: Previous ADC Results bits  
If ADPSIS = 1:  
Upper byte of FLTR at the start of current ADC conversion  
If ADPSIS = 0:  
Upper bits of ADRES at the start of current ADC conversion<sup>(1)</sup>

**Note 1:** If ADPSIS = 0, ADPREVH and ADPREVL are formatted the same way as ADRES is, depending on the FM bit.

## REGISTER 37-23: ADPREVL: ADC PREVIOUS RESULT REGISTER

|                                                |     |     |     |     |     |     |     |
|------------------------------------------------|-----|-----|-----|-----|-----|-----|-----|
| R-x                                            | R-x | R-x | R-x | R-x | R-x | R-x | R-x |
| PREV<7:0>                                      |     |     |     |     |     |     |     |
| bit 7 <span style="float: right;">bit 0</span> |     |     |     |     |     |     |     |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0 **PREV<7:0>**: Previous ADC Results bits  
If ADPSIS = 1:  
Lower byte of FLTR at the start of current ADC conversion  
If ADPSIS = 0:  
Lower bits of ADRES at the start of current ADC conversion<sup>(1)</sup>

**Note 1:** If ADPSIS = 0, ADPREVH and ADPREVL are formatted the same way as ADRES is, depending on the FM bit.

## REGISTER 37-24: ADACCU: ADC ACCUMULATOR REGISTER UPPER

|       |     |     |     |     |     |            |         |
|-------|-----|-----|-----|-----|-----|------------|---------|
| U-0   | U-0 | U-0 | U-0 | U-0 | U-0 | R/W-x/x    | R/W-x/x |
| —     | —   | —   | —   | —   | —   | ACC<17:16> |         |
| bit 7 |     |     |     |     |     |            | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-2

**Unimplemented:** Read as '0'

bit 1-0

**ACC<17:16>:** ADC Accumulator MSB. Upper two bits of accumulator value. See [Table 37-2](#) for more details.

## REGISTER 37-25: ADACCH: ADC ACCUMULATOR REGISTER HIGH

|           |         |         |         |         |         |         |         |
|-----------|---------|---------|---------|---------|---------|---------|---------|
| R/W-x/x   | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x |
| ACC<15:8> |         |         |         |         |         |         |         |
| bit 7     |         |         |         |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-0

**ACC<15:8>:** ADC Accumulator middle bits. Middle eight bits of accumulator value. See [Table 37-2](#) for more details.

## REGISTER 37-26: ADACCL: ADC ACCUMULATOR REGISTER LOW

|          |         |         |         |         |         |         |         |
|----------|---------|---------|---------|---------|---------|---------|---------|
| R/W-x/x  | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x |
| ACC<7:0> |         |         |         |         |         |         |         |
| bit 7    |         |         |         |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-0

**ACC<7:0>:** ADC Accumulator LSB. Lower eight bits of accumulator value. See [Table 37-2](#) for more details.

## REGISTER 37-27: ADSTPTH: ADC THRESHOLD SETPOINT REGISTER HIGH

|            |         |         |         |         |         |         |         |
|------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0    | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| STPT<15:8> |         |         |         |         |         |         |         |
| bit 7      |         |         |         |         |         |         | bit 0   |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0 **STPT<15:8>**: ADC Threshold Setpoint MSB. Upper byte of ADC threshold setpoint, depending on ADCALC, may be used to determine ERR, see [Register 37-29](#) for more details.

## REGISTER 37-28: ADSTPTL: ADC THRESHOLD SETPOINT REGISTER LOW

|           |         |         |         |         |         |         |         |
|-----------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0   | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| STPT<7:0> |         |         |         |         |         |         |         |
| bit 7     |         |         |         |         |         |         | bit 0   |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0 **STPT<7:0>**: ADC Threshold Setpoint LSB. Lower byte of ADC threshold setpoint, depending on ADCALC, may be used to determine ERR, see [Register 37-30](#) for more details.

## REGISTER 37-29: ADERRH: ADC SETPOINT ERROR REGISTER HIGH

|           |     |     |     |       |     |     |     |
|-----------|-----|-----|-----|-------|-----|-----|-----|
| R-x       | R-x | R-x | R-x | R-x   | R-x | R-x | R-x |
| ERR<15:8> |     |     |     |       |     |     |     |
| bit 7     |     |     |     | bit 0 |     |     |     |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0 **ERR<15:8>**: ADC Setpoint Error MSB. Upper byte of ADC Setpoint Error. Setpoint Error calculation is determined by CALC bits of ADCON3, see [Register 37-4](#) for more details.

## REGISTER 37-30: ADERRL: ADC SETPOINT ERROR LOW BYTE REGISTER

|          |     |     |     |       |     |     |     |
|----------|-----|-----|-----|-------|-----|-----|-----|
| R-x      | R-x | R-x | R-x | R-x   | R-x | R-x | R-x |
| ERR<7:0> |     |     |     |       |     |     |     |
| bit 7    |     |     |     | bit 0 |     |     |     |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0 **ERR<7:0>**: ADC Setpoint Error LSB. Lower byte of ADC Setpoint Error calculation is determined by CALC bits of ADCON3, see [Register 37-4](#) for more details.

## REGISTER 37-31: ADLTHH: ADC LOWER THRESHOLD HIGH BYTE REGISTER

|           |         |         |         |         |         |         |         |
|-----------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0   | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| LTH<15:8> |         |         |         |         |         |         |         |
| bit 7     |         |         |         | bit 0   |         |         |         |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0 **LTH<15:8>**: ADC Lower Threshold MSB. LTH and UTH are compared with ERR to set the ADUTHR and ADLTHR bits of ADSTAT. Depending on the setting of ADTMD, an interrupt may be triggered by the results of this comparison.

## REGISTER 37-32: ADLTHL: ADC LOWER THRESHOLD LOW BYTE REGISTER

|          |         |         |         |         |         |         |         |
|----------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0  | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| LTH<7:0> |         |         |         |         |         |         |         |
| bit 7    |         |         |         |         |         |         | bit 0   |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0 **LTH<7:0>**: ADC Lower Threshold LSB. LTH and UTH are compared with ERR to set the ADUTHR and ADLTHR bits of ADSTAT. Depending on the setting of ADTMD, an interrupt may be triggered by the results of this comparison.

## REGISTER 37-33: ADUTHH: ADC UPPER THRESHOLD HIGH BYTE REGISTER

|           |         |         |         |         |         |         |         |
|-----------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0   | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| UTH<15:8> |         |         |         |         |         |         |         |
| bit 7     |         |         |         |         |         |         | bit 0   |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0 **UTH<15:8>**: ADC Upper Threshold MSB. LTH and UTH are compared with ERR to set the ADUTHR and ADLTHR bits of ADSTAT. Depending on the setting of ADTMD, an interrupt may be triggered by the results of this comparison.

## REGISTER 37-34: ADUTHL: ADC UPPER THRESHOLD LOW BYTE REGISTER

|          |         |         |         |         |         |         |         |
|----------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0  | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| UTH<7:0> |         |         |         |         |         |         |         |
| bit 7    |         |         |         |         |         |         | bit 0   |

### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as '0'                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set     | '0' = Bit is cleared |                                                       |

bit 7-0 **UTH<7:0>**: ADC Upper Threshold LSB. LTH and UTH are compared with ERR to set the ADUTHR and ADLTHR bits of ADSTAT. Depending on the setting of ADTMD, an interrupt may be triggered by the results of this comparison.



## REGISTER 37-35: ADACT: ADC AUTO-CONVERSION TRIGGER CONTROL REGISTER

| U-0   | U-0 | U-0 | R/W-0/0  | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|-------|-----|-----|----------|---------|---------|---------|---------|
| —     | —   | —   | ACT<4:0> |         |         |         |         |
| bit 7 |     |     |          |         |         |         |         |
|       |     |     |          |         |         |         | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-5 **Unimplemented:** Read as '0'

bit 4-0 **ADACT<4:0>:** Auto-Conversion Trigger Select Bits

11111 = Reserved, do not use

- 
- 
- 

11110 = Reserved, do not use

11101 = Software write to ADPCH

11100 = Reserved, do not use

11011 = Software read of ADRESH

11010 = Software read of ADERRH

11001 = CLC4\_out

11000 = CLC3\_out

10111 = CLC2\_out

10110 = CLC1\_out

10101 = Logical OR of all Interrupt-on-Change Interrupt Flags

10100 = CMP2\_out

10011 = CMP1\_out

10010 = NCO1\_out

10001 = PWM8\_out

10000 = PWM7\_out

01111 = PWM6\_out

01110 = PWM5\_out

01101 = CCP4\_trigger

01100 = CCP3\_trigger

01011 = CCP2\_trigger

01010 = CCP1\_trigger

01001 = SMT1\_trigger

01000 = TMR6\_postscaled

00111 = TMR5\_overflow

00110 = TMR4\_postscaled

00101 = TMR3\_overflow

00100 = TMR2\_postscaled

00011 = TMR1\_overflow

00010 = TMR0\_overflow

00001 = Pin selected by ADACTPPS

00000 = External Trigger Disabled

## REGISTER 37-36: ADCP: ADC CHARGE PUMP CONTROL REGISTER

|         |     |     |     |     |     |     |       |
|---------|-----|-----|-----|-----|-----|-----|-------|
| R/W-0/0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R-0/0 |
| CPON    | —   | —   | —   | —   | —   | —   | CPRDY |
| bit 7   |     |     |     |     |     |     | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

HS= Hardware set

bit 7

**CPON:** Charge Pump On Control bit

1 = Charge Pump On when requested by the ADC

0 = Charge Pump Off

bit 6-1

**Unimplemented:** Read as '0'

bit 0

**CPRDY:** Charge Pump Ready Status bit

1 = Charge Pump is ready

0 = Charge Pump is not ready (or never started)

**TABLE 37-6: SUMMARY OF REGISTERS ASSOCIATED WITH ADC**

| Name    | Bit 7       | Bit 6       | Bit 5      | Bit 4      | Bit 3       | Bit 2      | Bit 1       | Bit 0  | Register on Page |
|---------|-------------|-------------|------------|------------|-------------|------------|-------------|--------|------------------|
| ADCON0  | ON          | CONT        | —          | CS         | —           | FM         | —           | GO     | 671              |
| ADCON1  | ADPPOL      | ADIPEN      | ADGPOL     | —          | —           | —          | —           | ADDSEN | 672              |
| ADCON2  | ADPSIS      | ADCRS<2:0>  |            |            | ADACLR      | MD<2:0>    |             |        | 673              |
| ADCON3  | —           | ADCALC<2:0> |            |            | ADSOI       | ADTMD<2:0> |             |        | 674              |
| ADSTAT  | ADAOV       | ADUTHR      | ADLTHR     | ADMATH     | ADSTAT<3:0> |            |             |        | 675              |
| ADCLK   | —           | —           | CS<5:0>    |            |             |            |             |        | 676              |
| ADREF   | —           | —           | —          | ADNREF     | —           | —          | ADPREF<1:0> |        | 676              |
| ADPCH   | —           | —           | ADPCH<5:0> |            |             |            |             |        | 677              |
| ADPREL  | PRE<7:0>    |             |            |            |             |            |             |        | 678              |
| ADPREH  | —           | —           | —          | PRE<12:8>  |             |            |             |        | 678              |
| ADACQL  | ACQ<7:0>    |             |            |            |             |            |             |        | 679              |
| ADCAP   | —           | —           | —          | ADCAP<4:0> |             |            |             |        | 680              |
| ADRPT   | RPT<7:0>    |             |            |            |             |            |             |        | 680              |
| ADCNT   | CNT<7:0>    |             |            |            |             |            |             |        | 681              |
| ADFLTRL | FLTR<7:0>   |             |            |            |             |            |             |        | 681              |
| ADFLTRH | FLTR<15:8>  |             |            |            |             |            |             |        | 681              |
| ADRESL  | ADRESL<7:0> |             |            |            |             |            |             |        | 682, 683         |
| ADRESH  | ADRESH<7:0> |             |            |            |             |            |             |        | 682, 683         |
| ADPREVH | PREV<15:8>  |             |            |            |             |            |             |        | 684              |
| ADPREVL | PREV<7:0>   |             |            |            |             |            |             |        | 684              |
| ADACCH  | ACC<15:8>   |             |            |            |             |            |             |        | 685              |
| ADACCL  | ACC<7:0>    |             |            |            |             |            |             |        | 685              |
| ADACCU  | —           | —           | —          | —          | —           | —          | ACC<17:16>  |        | 685              |
| ADSTPTL | STPT<7:0>   |             |            |            |             |            |             |        | 686              |
| ADSTPTH | STPT<15:8>  |             |            |            |             |            |             |        | 686              |
| ADERRL  | ERR<7:0>    |             |            |            |             |            |             |        | 687              |
| ADERRH  | ERR<15:8>   |             |            |            |             |            |             |        | 687              |
| ADLTHH  | LTH<15:8>   |             |            |            |             |            |             |        | 687              |
| ADLTHL  | LTH<7:0>    |             |            |            |             |            |             |        | 688              |
| ADUTHH  | UTH<15:8>   |             |            |            |             |            |             |        | 688              |
| ADUTHL  | UTH<7:0>    |             |            |            |             |            |             |        | 688              |
| ADERRL  | ERR<15:8>   |             |            |            |             |            |             |        | 687              |
| ADACT   | —           | —           | —          | —          | ADACT<4:0>  |            |             |        | 689              |
| ADCP    | CPON        | —           | —          | —          | —           | —          | —           | CPRDY  | 690              |

**Legend:** — = unimplemented read as '0'. Shaded cells are not used for the ADC module.

## 38.0 5-BIT DIGITAL-TO-ANALOG CONVERTER (DAC) MODULE

The Digital-to-Analog Converter supplies a variable voltage reference, ratiometric with the input source, with 32 selectable output levels.

The positive input source ( $V_{SOURCE+}$ ) of the DAC can be connected to:

- FVR Buffer
- External  $V_{REF+}$  pin
- $V_{DD}$  supply voltage

The negative input source ( $V_{SOURCE-}$ ) of the DAC can be connected to:

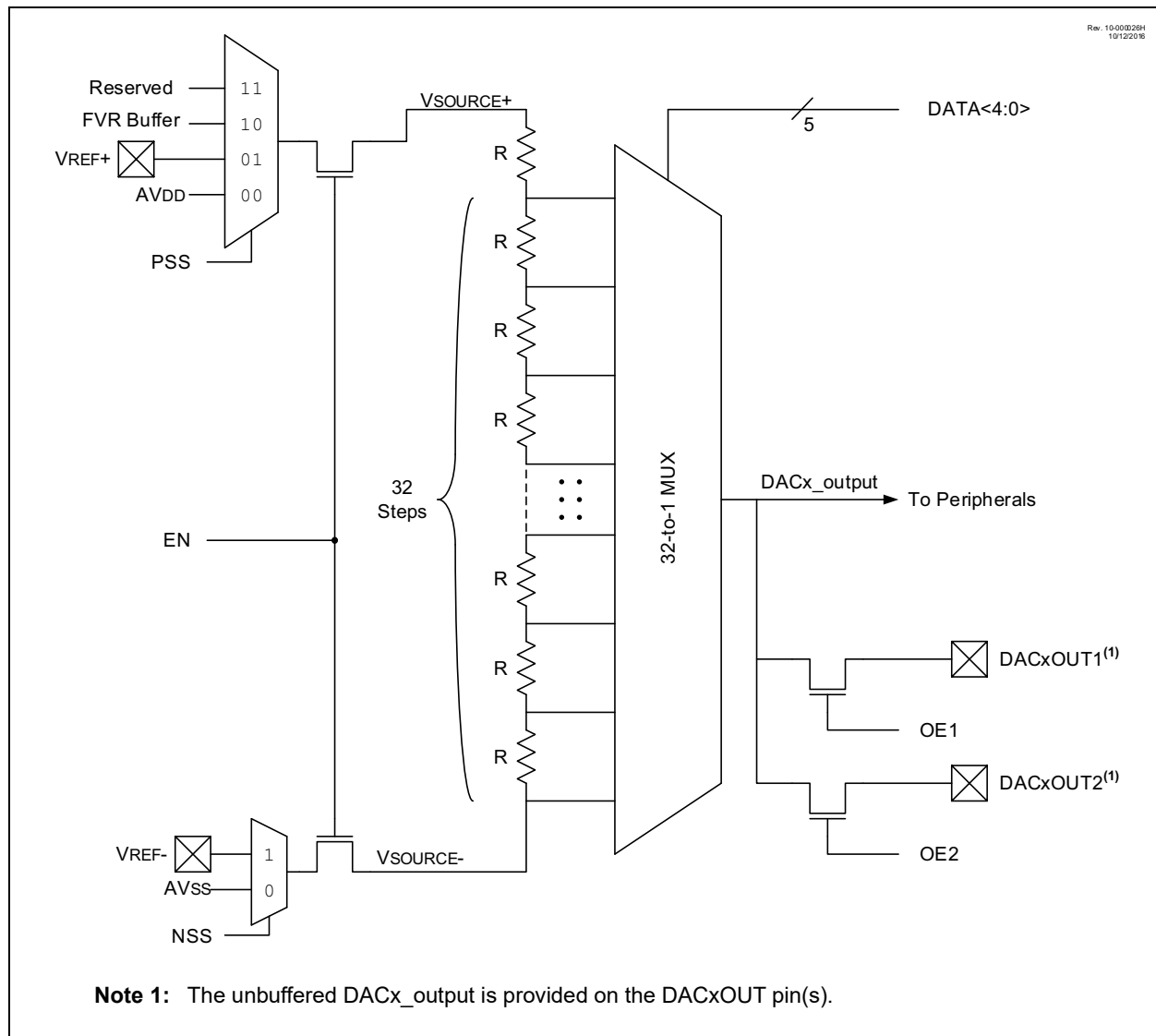
- External  $V_{REF-}$  pin
- $V_{SS}$

The output of the DAC ( $DAC1\_output$ ) can be selected as a reference voltage to the following:

- Comparator positive input
- ADC input channel
- $DAC1OUT1$  pin
- $DAC1OUT2$  pin

The Digital-to-Analog Converter (DAC) can be enabled by setting the EN bit of the  $DAC1CON0$  register.

**FIGURE 38-1: DIGITAL-TO-ANALOG CONVERTER BLOCK DIAGRAM**



## 38.1 Output Voltage Selection

The DAC has 32 voltage level ranges. The 32 levels are set with the DATA<4:0> bits of the DAC1CON1 register.

The DAC output voltage can be determined by using [Equation 38-1](#).

## 38.2 Ratiometric Output Level

The DAC output value is derived using a resistor ladder with each end of the ladder tied to a positive and negative voltage reference input source. If the voltage of either input source fluctuates, a similar fluctuation will result in the DAC output value.

The value of the individual resistors within the ladder can be found in [Table 45-16](#).

## 38.3 DAC Voltage Reference Output

The unbuffered DAC voltage can be output to the DAC1OUTn pin(s) by setting the respective DACOEn bit(s) of the DAC1CON0 register. Selecting the DAC reference voltage for output on either DAC1OUTn pin automatically overrides the digital output buffer, the weak pull-up and digital input threshold detector functions of that pin.

Reading the DAC1OUTn pin when it has been configured for DAC reference voltage output will always return a '0'.

**Note:** The unbuffered DAC output (DAC1OUTn) is not intended to drive an external load.

## 38.4 Operation During Sleep

When the device wakes up from Sleep through an interrupt or a Windowed Watchdog Timer Time-out, the contents of the DAC1CON0 register are not affected. To minimize current consumption in Sleep mode, the voltage reference should be disabled.

## 38.5 Effects of a Reset

A device Reset affects the following:

- DAC1 is disabled.
- DAC1 output voltage is removed from the DAC1OUTn pin(s).
- The DAC1R<4:0> range select bits are cleared.

### EQUATION 38-1: DAC OUTPUT VOLTAGE

**IF DACEN = 1**

$$\text{DACx\_output} = \left( (V_{\text{REF+}} - V_{\text{REF-}}) \times \frac{\text{DATA}[4:0]}{2^5} \right) + V_{\text{REF-}}$$

**Note:** See the DAC1CON0 register for the available VSOURCE+ and VSOURCE- selections.

## 38.6 Register Definitions: DAC Control

Long bit name prefixes for the DAC peripheral is shown below. Refer to [Section 1.3.2.2 “Long Bit Names”](#) for more information.

| Peripheral | Bit Name Prefix |
|------------|-----------------|
| DAC1       | DAC1            |

### REGISTER 38-1: DAC1CON0: DAC CONTROL REGISTER

|         |     |         |         |          |         |     |         |
|---------|-----|---------|---------|----------|---------|-----|---------|
| R/W-0/0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0  | R/W-0/0 | U-0 | R/W-0/0 |
| EN      | —   | OE1     | OE2     | PSS<1:0> | —       | NSS |         |
| bit 7   |     |         |         |          |         |     | bit 0   |

#### Legend:

|                      |                      |                                                       |
|----------------------|----------------------|-------------------------------------------------------|
| R = Readable bit     | W = Writable bit     | U = Unimplemented bit, read as ‘0’                    |
| u = Bit is unchanged | x = Bit is unknown   | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set     | ‘0’ = Bit is cleared |                                                       |

|         |                                                                                                                                                                 |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| bit 7   | <b>EN:</b> DAC Enable bit<br>1 = DAC is enabled<br>0 = DAC is disabled <sup>(1)</sup>                                                                           |
| bit 6   | <b>Unimplemented:</b> Read as ‘0’                                                                                                                               |
| bit 5   | <b>OE1:</b> DAC Voltage Output Enable bit<br>1 = DAC voltage level is output on the DAC1OUT1 pin<br>0 = DAC voltage level is disconnected from the DAC1OUT1 pin |
| bit 4   | <b>OE2:</b> DAC Voltage Output Enable bit<br>1 = DAC voltage level is output on the DAC1OUT2 pin<br>0 = DAC voltage level is disconnected from the DAC1OUT2 pin |
| bit 3-2 | <b>PSS&lt;1:0&gt;:</b> DAC Positive Source Select bit<br>11 = Reserved<br>10 = FVR buffer 2<br>01 = VREF+<br>00 = VDD                                           |
| bit 1   | <b>Unimplemented:</b> Read as ‘0’                                                                                                                               |
| bit 0   | <b>NSS:</b> DAC Negative Source Select bit<br>1 = VREF-<br>0 = VSS                                                                                              |

**Note 1:** DAC1OUTx output pins are still active.

**REGISTER 38-2: DAC1CON1: DAC DATA REGISTER**

|       |     |     |           |         |         |         |         |
|-------|-----|-----|-----------|---------|---------|---------|---------|
| U-0   | U-0 | U-0 | R/W-0/0   | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| —     | —   | —   | DATA<4:0> |         |         |         |         |
| bit 7 |     |     | bit 0     |         |         |         |         |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-5

**Unimplemented:** Read as '0'

bit 4-0

**DATA<4:0>:** Data Input Register for DAC bits

**TABLE 38-1: SUMMARY OF REGISTERS ASSOCIATED WITH THE DAC MODULE**

| Name     | Bit 7 | Bit 6 | Bit 5 | Bit 4     | Bit 3    | Bit 2 | Bit 1 | Bit 0 | Register on page    |
|----------|-------|-------|-------|-----------|----------|-------|-------|-------|---------------------|
| DAC1CON0 | EN    | —     | OE1   | OE2       | PSS<1:0> | —     | NSS   |       | <a href="#">694</a> |
| DAC1CON1 | —     | —     | —     | DATA<4:0> |          |       |       |       | <a href="#">695</a> |

**Legend:** — = Unimplemented location, read as '0'. Shaded cells are not used with the DAC module.

## 39.0 COMPARATOR MODULE

**Note:** The PIC18(L)F25/26K83 devices have two comparators. Therefore, all information in this section refers to both C1 and C2.

Comparators are used to interface analog circuits to a digital circuit by comparing two analog voltages and providing a digital indication of their relative magnitudes. Comparators are very useful mixed signal building blocks because they provide analog functionality independent of program execution.

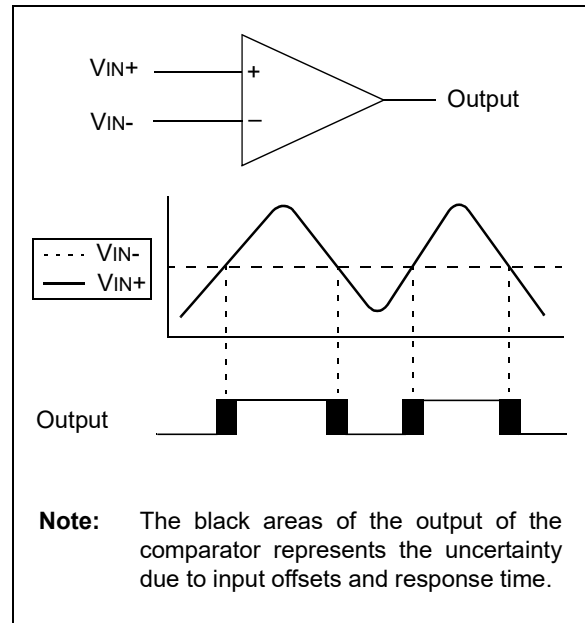
The analog comparator module includes the following features:

- Programmable input selection
- Programmable output polarity
- Rising/falling output edge interrupts

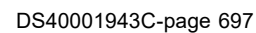
## 39.1 Comparator Overview

A single comparator is shown in Figure 39-1 along with the relationship between the analog input levels and the digital output. When the analog voltage at  $V_{IN+}$  is less than the analog voltage at  $V_{IN-}$ , the output of the comparator is a digital low level. When the analog voltage at  $V_{IN+}$  is greater than the analog voltage at  $V_{IN-}$ , the output of the comparator is a digital high level.

**FIGURE 39-1: SINGLE COMPARATOR**







## 39.2 Comparator Control

Each comparator has two control registers: CMxCON0 and CMxCON1.

The CMxCON0 register (see [Register 39-1](#)) contains Control and Status bits for the following:

- Enable
- Output
- Output polarity
- Hysteresis enable
- Timer1 output synchronization

The CMxCON1 register (see [Register 39-2](#)) contains Control bits for the following:

- Interrupt on positive/negative edge enables

The CMxPCH and CMxNCH registers are used to select the positive and negative input channels, respectively.

### 39.2.1 COMPARATOR ENABLE

Setting the EN bit of the CMxCON0 register enables the comparator for operation. Clearing the EN bit disables the comparator resulting in minimum current consumption.

### 39.2.2 COMPARATOR OUTPUT

The output of the comparator can be monitored by reading either the CxOUT bit of the CMxCON0 register or the CxOUT bit of the CMOUT register.

The comparator output can also be routed to an external pin through the RxyPPS register ([Register 17-2](#)). The corresponding TRIS bit must be clear to enable the pin as an output.

**Note 1:** The internal output of the comparator is latched with each instruction cycle. Unless otherwise specified, external outputs are not latched.

### 39.2.3 COMPARATOR OUTPUT POLARITY

Inverting the output of the comparator is functionally equivalent to swapping the comparator inputs. The polarity of the comparator output can be inverted by setting the POL bit of the CMxCON0 register. Clearing the POL bit results in a noninverted output.

[Table 39-1](#) shows the output state versus input conditions, including polarity control.

**TABLE 39-1: COMPARATOR OUTPUT STATE VS. INPUT CONDITIONS**

| Input Condition | POL | CxOUT |
|-----------------|-----|-------|
| $CxVN > CxVP$   | 0   | 0     |
| $CxVN < CxVP$   | 0   | 1     |
| $CxVN > CxVP$   | 1   | 1     |
| $CxVN < CxVP$   | 1   | 0     |

## 39.3 Comparator Hysteresis

A selectable amount of separation voltage can be added to the input pins of each comparator to provide a hysteresis function to the overall operation. Hysteresis is enabled by setting the HYS bit of the CMxCON0 register.

See Comparator Specifications in [Table 45-15](#) for more information.

### 39.3.1 COMPARATOR OUTPUT SYNCHRONIZATION

The output from a comparator can be synchronized with Timer1 by setting the SYNC bit of the CMxCON0 register.

Once enabled, the comparator output is latched on the falling edge of the Timer1 source clock. If a prescaler is used, the CxOUT bit is synchronized with the timer, so that the software sees no ambiguity due to timing. See the Comparator Block Diagram ([Figure 39-2](#)) and the Timer1 Block Diagram ([Figure 21-1](#)) for more information.

## 39.4 Comparator Interrupt

An interrupt can be generated for every rising or falling edge of the comparator output.

When either edge detector is triggered and its associated enable bit is set (INTP and/or INTN bits of the CMxCON1 register), the Corresponding Interrupt Flag bit (CxIF bit of the respective PIR register) will be set.

To enable the interrupt, you must set the following bits:

- EN bit of the CMxCON0 register
- CxIE bit of the respective PIE register
- INTP bit of the CMxCON1 register (for a rising edge detection)
- INTN bit of the CMxCON1 register (for a falling edge detection)
- GIE bit of the INTCON0 register

The associated interrupt flag bit, CxIF bit of the respective PIR register, must be cleared in software. If another edge is detected while this flag is being cleared, the flag will still be set at the end of the sequence.

**Note:** Although a comparator is disabled, an interrupt can be generated by changing the output polarity with the POL bit of the CMxCON0 register, or by switching the comparator on or off with the EN bit of the CMxCON0 register.

## 39.5 Comparator Positive Input Selection

Configuring the PCH<2:0> bits of the CMxPCH register directs an internal voltage reference or an analog pin to the non-inverting input of the comparator:

- CxIN0+, CxIN1+ analog pin
- DAC output
- FVR (Fixed Voltage Reference)
- Vss (Ground)

See [Section 35.0 “Fixed Voltage Reference \(FVR\)”](#) for more information on the Fixed Voltage Reference module.

See [Section 38.0 “5-Bit Digital-to-Analog Converter \(DAC\) Module”](#) for more information on the DAC input signal.

Any time the comparator is disabled (EN = 0), all comparator inputs are disabled.

## 39.6 Comparator Negative Input Selection

The NCH<2:0> bits of the CMxNCH register direct an analog input pin and internal reference voltage or analog ground to the inverting input of the comparator:

- CxIN0-, CxIN1-, CxIN2-, CxIN3- analog pin
- FVR (Fixed Voltage Reference)
- Analog Ground

**Note:** To use CxINy+ and CxINy- pins as analog input, the appropriate bits must be set in the ANSEL register and the corresponding TRIS bits must also be set to disable the output drivers.

## 39.7 Comparator Response Time

The comparator output is indeterminate for a period of time after the change of an input source or the selection of a new reference voltage. This period is referred to as the response time. The response time of the comparator differs from the settling time of the voltage reference. Therefore, both of these times must be considered when determining the total response time to a comparator input change. See the Comparator and Voltage Reference Specifications in [Table 45-15](#) and [Table 45-17](#) for more details.

## 39.8 Analog Input Connection Considerations

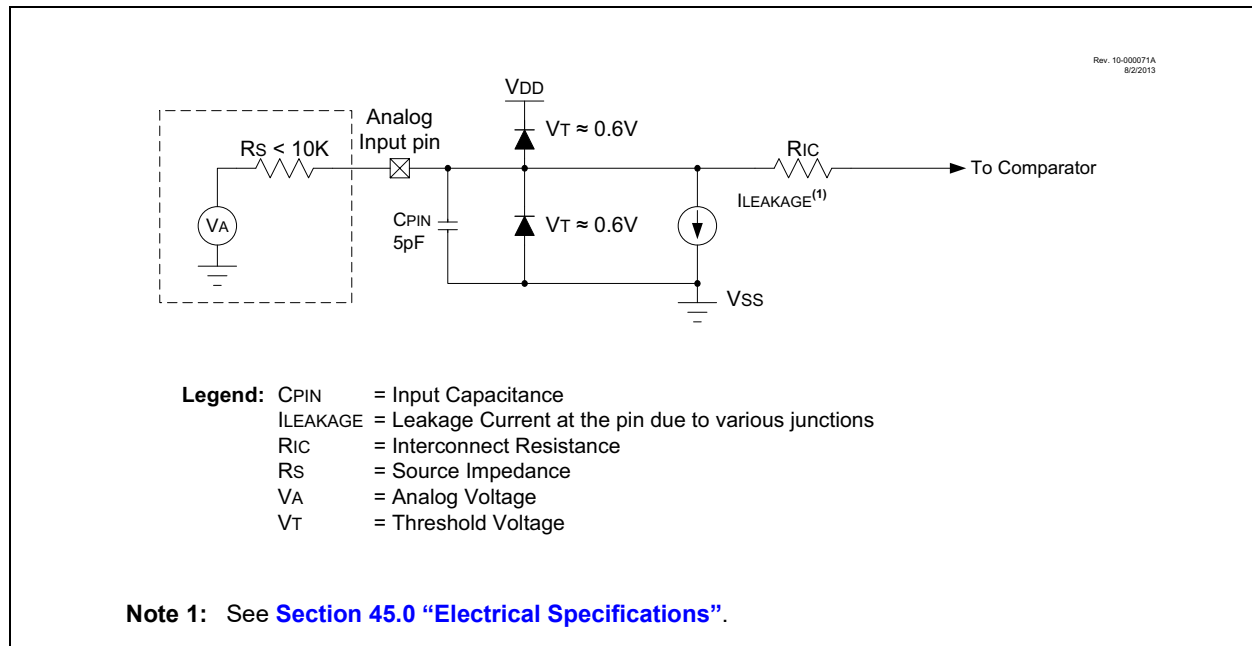
A simplified circuit for an analog input is shown in [Figure 39-3](#). Since the analog input pins share their connection with a digital input, they have reverse biased ESD protection diodes to  $V_{DD}$  and  $V_{SS}$ . The analog input, therefore, must be between  $V_{SS}$  and  $V_{DD}$ . If the input voltage deviates from this range by more than 0.6V in either direction, one of the diodes is forward biased and a latch-up may occur.

A maximum source impedance of 10 k $\Omega$  is recommended for the analog sources. Also, any external component connected to an analog input pin, such as a capacitor or a Zener diode, should have very little leakage current to minimize inaccuracies introduced.

**Note 1:** When reading a PORT register, all pins configured as analog inputs will read as a '0'. Pins configured as digital inputs will convert as an analog input, according to the input specification.

**2:** Analog levels on any pin defined as a digital input, may cause the input buffer to consume more current than is specified.

**FIGURE 39-3: ANALOG INPUT MODEL**



## 39.9 CWG1 Auto-Shutdown Source

The output of the comparator module can be used as an auto-shutdown source for the CWG1 module. When the output of the comparator is active and the corresponding WGASxE is enabled, the CWG operation will be suspended immediately (see [Section 26.10.1.2 “External Input Source”](#)).

## 39.10 ADC Auto-Trigger Source

The output of the comparator module can be used to trigger an ADC conversion. When the ADACT register is set to trigger on a comparator output, an ADC conversion will trigger when the Comparator output goes high.

## 39.11 TMR2/4/6 Reset

The output of the comparator module can be used to reset Timer2. When the TxERS register is appropriately set, the timer will reset when the Comparator output goes high.

## 39.12 Operation in Sleep Mode

The comparator module can operate during Sleep. The comparator clock source is based on the Timer1 clock source. If the Timer1 clock source is either the system clock (FOSC) or the instruction clock (FOSC/4), Timer1 will not operate during Sleep, and synchronized comparator outputs will not operate.

A comparator interrupt will wake the device from Sleep. The CxIE bits of the respective PIE register must be set to enable comparator interrupts.

## 39.13 Register Definitions: Comparator Control

Long bit name prefixes for the Comparators are shown in Table 39-2. Refer to Section 1.3.2.2 “Long Bit Names” for more information.

TABLE 39-2:

| Peripheral | Bit Name Prefix |
|------------|-----------------|
| C1         | C1              |
| C2         | C2              |

REGISTER 39-1: CMxCON0: COMPARATOR x CONTROL REGISTER 0

|         |       |     |         |     |     |         |         |
|---------|-------|-----|---------|-----|-----|---------|---------|
| R/W-0/0 | R-0/0 | U-0 | R/W-0/0 | U-0 | U-1 | R/W-0/0 | R/W-0/0 |
| EN      | OUT   | —   | POL     | —   | —   | HYS     | SYNC    |
| bit 7   |       |     |         |     |     |         | bit 0   |

### Legend:

R = Readable bit                      W = Writable bit                      U = Unimplemented bit, read as ‘0’  
 -n = Value at POR                      ‘1’ = Bit is set                      ‘0’ = Bit is cleared                      x = Bit is unknown

- bit 7                      **EN:** Comparator Enable bit  
                             1 = Comparator is enabled  
                             0 = Comparator is disabled and consumes no active power
- bit 6                      **OUT:** Comparator Output bit  
                             If POL = 0 (noninverted polarity):  
                             1 = CxVP > CxVN  
                             0 = CxVP < CxVN  
                             If POL = 1 (inverted polarity):  
                             1 = CxVP < CxVN  
                             0 = CxVP > CxVN
- bit 5                      **Unimplemented:** Read as ‘0’
- bit 4                      **POL:** Comparator Output Polarity Select bit  
                             1 = Comparator output is inverted  
                             0 = Comparator output is not inverted
- bit 3                      **Unimplemented:** Read as ‘0’
- bit 2                      **Unimplemented:** Read as ‘1’
- bit 1                      **HYS:** Comparator Hysteresis Enable bit  
                             1 = Comparator hysteresis enabled  
                             0 = Comparator hysteresis disabled
- bit 0                      **SYNC:** Comparator Output Synchronous Mode bit  
                             1 = Comparator output to Timer1/3/5 and I/O pin is synchronous to changes on Timer1 clock source.  
                             0 = Comparator output to Timer1/3/5 and I/O pin is asynchronous  
                                 Output updated on the falling edge of Timer1/3/5 clock source.

## REGISTER 39-2: CMxCON1: COMPARATOR x CONTROL REGISTER 1

|       |     |     |     |     |     |         |         |
|-------|-----|-----|-----|-----|-----|---------|---------|
| U-0   | U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 |
| —     | —   | —   | —   | —   | —   | INTP    | INTN    |
| bit 7 |     |     |     |     |     | bit 0   |         |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-2 Unimplemented: Read as '0'

bit 1 **INTP**: Comparator Interrupt on Positive-Going Edge Enable bit

1 = The CxIF interrupt flag will be set upon a positive-going edge of the CxOUT bit

0 = No interrupt flag will be set on a positive-going edge of the CxOUT bit

bit 0 **INTN**: Comparator Interrupt on Negative-Going Edge Enable bit

1 = The CxIF interrupt flag will be set upon a negative-going edge of the CxOUT bit

0 = No interrupt flag will be set on a negative-going edge of the CxOUT bit

## REGISTER 39-3: CMxNCH: COMPARATOR x INVERTING CHANNEL SELECT REGISTER

|       |     |     |     |     |          |         |         |
|-------|-----|-----|-----|-----|----------|---------|---------|
| U-0   | U-0 | U-0 | U-0 | U-0 | R/W-0/0  | R/W-0/0 | R/W-0/0 |
| —     | —   | —   | —   | —   | NCH<2:0> |         |         |
| bit 7 |     |     |     |     |          | bit 0   |         |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-3 **Unimplemented**: Read as '0'

bit 2-0 **NCH<2:0>**: Comparator Inverting Input Channel Select bits

111 = Vss

110 = FVR\_Buffer2

101 = NCH not connected

100 = NCH not connected

011 = CxIN3-

010 = CxIN2-

001 = CxIN1-

000 = CxIN0-

**REGISTER 39-4: CMxPCH: COMPARATOR x NON-INVERTING CHANNEL SELECT REGISTER**

|       |     |     |     |     |          |         |         |
|-------|-----|-----|-----|-----|----------|---------|---------|
| U-0   | U-0 | U-0 | U-0 | U-0 | R/W-0/0  | R/W-0/0 | R/W-0/0 |
| —     | —   | —   | —   | —   | PCH<2:0> |         |         |
| bit 7 |     |     |     |     |          |         | bit 0   |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-3 **Unimplemented:** Read as '0'

bit 2-0 **PCH<2:0>:** Comparator Non-Inverting Input Channel Select bits

111 = Vss

110 = FVR\_Buffer2

101 = DAC\_Output

100 = PCH not connected

011 = PCH not connected

010 = PCH not connected

001 = CxIN1+

000 = CxIN0+

**REGISTER 39-5: CMOUT: COMPARATOR OUTPUT REGISTER**

|       |     |     |     |     |     |       |       |
|-------|-----|-----|-----|-----|-----|-------|-------|
| U-0   | U-0 | U-0 | U-0 | U-0 | U-0 | R-0/0 | R-0/0 |
| —     | —   | —   | —   | —   | —   | C2OUT | C1OUT |
| bit 7 |     |     |     |     |     |       | bit 0 |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-2 **Unimplemented:** Read as '0'

bit 1 **C2OUT:** Mirror copy of C2OUT bit

bit 0 **C1OUT:** Mirror copy of C1OUT bit

**TABLE 39-3: SUMMARY OF REGISTERS ASSOCIATED WITH COMPARATOR MODULE**

| Name    | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2    | Bit 1 | Bit 0 | Reset Values on page |
|---------|-------|-------|-------|-------|-------|----------|-------|-------|----------------------|
| CMxCON0 | EN    | OUT   | —     | POL   | —     | —        | HYS   | SYNC  | <a href="#">702</a>  |
| CMxCON1 | —     | —     | —     | —     | —     | —        | INTP  | INTN  | <a href="#">703</a>  |
| CMxNCH  | —     | —     | —     | —     | —     | NCH<2:0> |       |       | <a href="#">703</a>  |
| CMxPCH  | —     | —     | —     | —     | —     | PCH<2:0> |       |       | <a href="#">704</a>  |
| CMOUT   | —     | —     | —     | —     | —     | —        | C2OUT | C1OUT | <a href="#">704</a>  |

**Legend:** — = unimplemented, read as '0'. Shaded cells are unused by the comparator module.



## 40.0 HIGH/LOW-VOLTAGE DETECT (HLVD)

The PIC18(L)F25/26K83 family of devices has a High/Low-Voltage Detect module (HLVD). This is a programmable circuit that sets both a device voltage trip point and the direction of change from that point (positive going, negative going or both). If the device experiences an excursion past the trip point in that direction, an interrupt flag is set. If the interrupt is enabled, the program execution branches to the interrupt vector address and the software responds to the interrupt.

Complete control of the HLVD module is provided through the HLVDCON0 and HLVDCON1 register. This allows the circuitry to be “turned off” by the user under software control, which minimizes the current consumption for the device.

The module’s block diagram is shown in [Figure 40-1](#).

Since the HLVD can be software enabled through the EN bit, setting and clearing the enable bit does not produce a false HLVD event glitch. Each time the HLVD module is enabled, the circuitry requires some time to stabilize. The RDY bit (HLVDCON0<4>) is a read-only bit used to indicate when the band gap reference voltages are stable.

The module can only generate an interrupt after the module is turned ON and the band gap reference voltages are ready.

The INTH and INTL bits determine the overall operation of the module. When INTH is set, the module monitors for rises in VDD above the trip point set by the HLVDCON1 register. When INTL is set, the module monitors for drops in VDD below the trip point set by the HLVDCON1 register. When both the INTH and INTL bits are set, any changes above or below the trip point set by the HLVDCON1 register can be monitored.

The OUT bit can be read to determine if the voltage is greater than or less than the voltage level selected by the HLVDCON1 register.

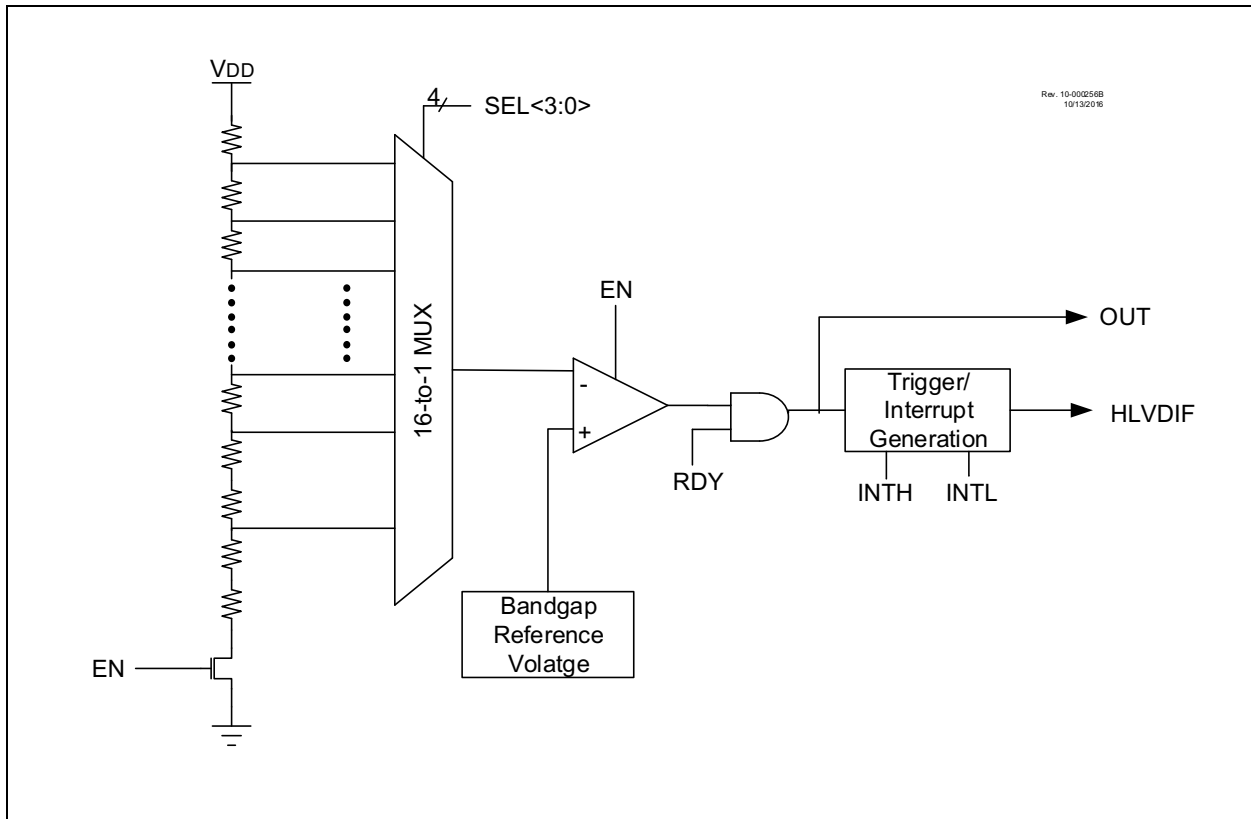
## 40.1 Operation

When the HLVD module is enabled, a comparator uses an internally generated voltage reference as the set point. The set point is compared with the trip point, where each node in the resistor divider represents a trip point voltage. The “trip point” voltage is the voltage level at which the device detects a high or low-voltage event, depending on the configuration of the module.

When the supply voltage is equal to the trip point, the voltage tapped off of the resistor array is equal to the internal reference voltage generated by the voltage reference module. The comparator then generates an interrupt signal by setting the HLVDIF bit.

The trip point voltage is software programmable to any of SEL<3:0> bits (HLVDCON1<3:0>).

**FIGURE 40-1: HLVD MODULE BLOCK DIAGRAM**



## 40.2 HLVD Setup

To set up the HLVD module:

1. Select the desired HLVD trip point by writing the value to the SEL<3:0> bits of the HLVDCON1 register.
2. Depending on the application to detect high-voltage peaks or low-voltage drops or both, set the INTH or INTL bit appropriately.
3. Enable the HLVD module by setting the EN bit.
4. Clear the HLVD interrupt flag (PIR2 register), which may have been set from a previous interrupt.
5. If interrupts are desired, enable the HLVD interrupt by setting the HLVDIE in the PIE2 register and GIE bits.

An interrupt will not be generated until the RDY bit is set.

**Note:** Before changing any module settings (INTH, INTL, SEL<3:0>), first disable the module (EN = 0), make the changes and re-enable the module. This prevents the generation of false HLVD events.

## 40.3 Current Consumption

When the module is enabled, the HLVD comparator and voltage divider are enabled and consume static current. The total current consumption, when enabled, is specified in electrical specification Parameter **D206** (Table 45-3).

Depending on the application, the HLVD module does not need to operate constantly. To reduce current requirements, the HLVD circuitry may only need to be enabled for short periods where the voltage is checked. After such a check, the module could be disabled.

## 40.4 HLVD Start-up Time

The internal reference voltage of the HLVD module, specified in electrical specification (Table 45-17), may be used by other internal circuitry, such as the programmable Brown-out Reset. If the HLVD or other circuits using the voltage reference are disabled to lower the device's current consumption, the reference voltage circuit will require time to become stable before a low or high-voltage condition can be reliably detected. This start-up time,  $T_{FVRST}$ , is an interval that is independent of device clock speed. It is specified in electrical specification (Table 45-17).

The HLVD interrupt flag is not enabled until  $T_{FVRST}$  has expired and a stable reference voltage is reached. For this reason, brief excursions beyond the set point may not be detected during this interval (see Figure 40-2 or Figure 40-3).

**FIGURE 40-2: LOW-VOLTAGE DETECT OPERATION (INTL = 1)**

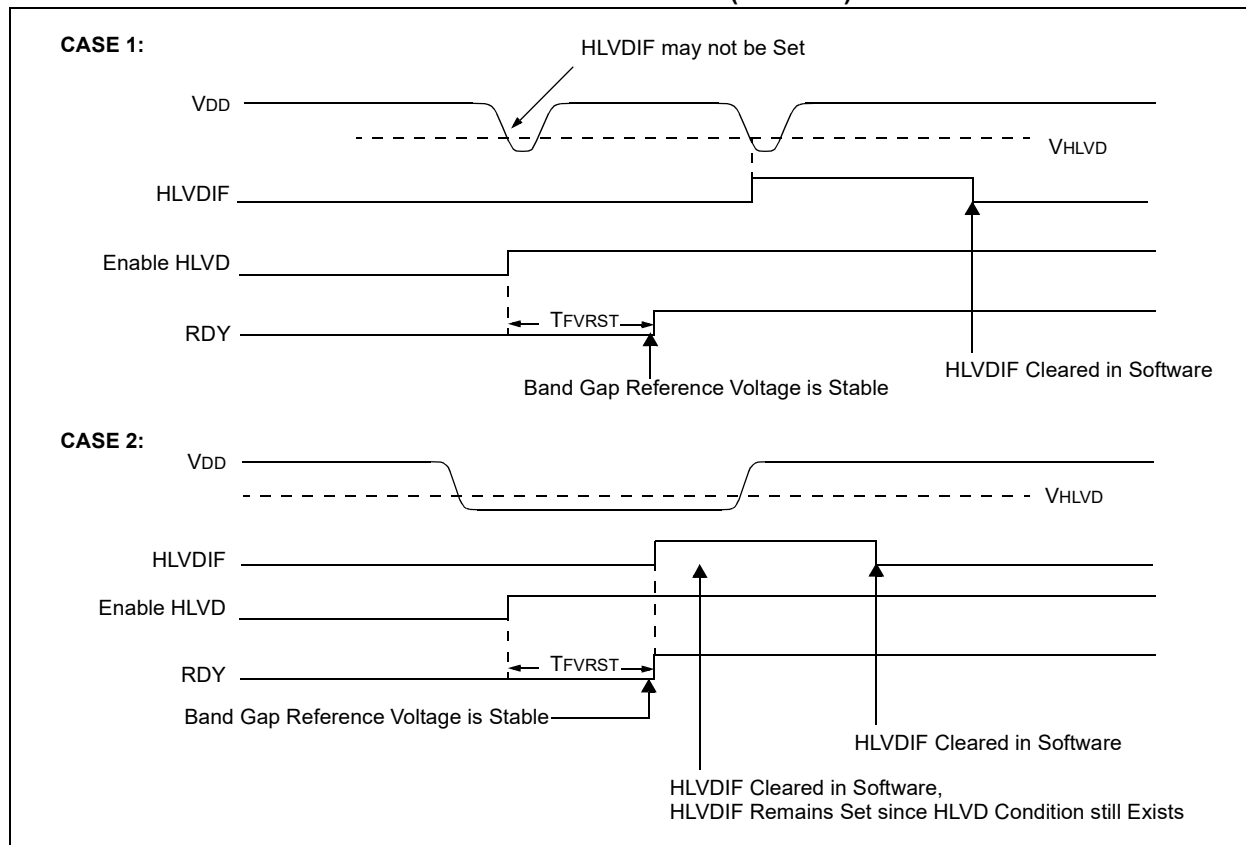
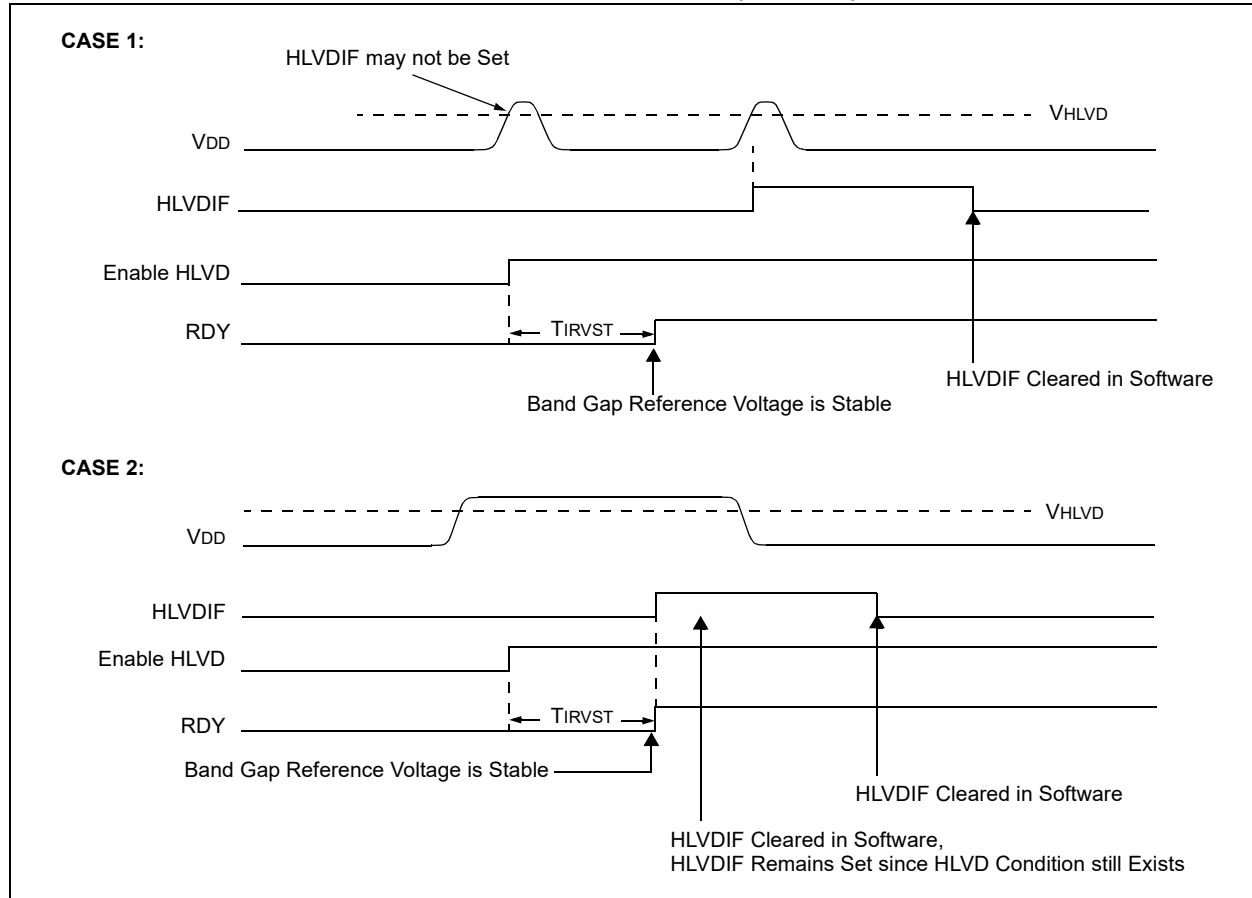


**FIGURE 40-3: HIGH-VOLTAGE DETECT OPERATION (INTH = 1)**

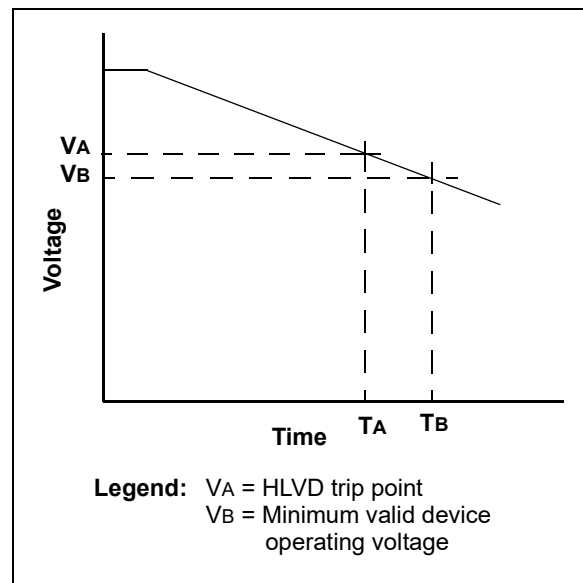


## 40.5 Applications

In many applications, it is desirable to detect a drop below, or rise above, a particular voltage threshold. For example, the HLVD module could be periodically enabled to detect Universal Serial Bus (USB) attach or detach. This assumes the device is powered by a lower voltage source than the USB when detached. An attach would indicate a High-Voltage Detect from, for example, 3.3V to 5V (the voltage on USB) and vice versa for a detach. This feature could save a design a few extra components and an attach signal (input pin).

For general battery applications, Figure 40-4 shows a possible voltage curve. Over time, the device voltage decreases. When the device voltage reaches voltage, V<sub>A</sub>, the HLVD logic generates an interrupt at time, T<sub>A</sub>. The interrupt could cause the execution of an Interrupt Service Routine (ISR), which would allow the application to perform "housekeeping tasks" and a controlled shutdown before the device voltage exits the valid operating range at T<sub>B</sub>. This would give the application a time window, represented by the difference between T<sub>A</sub> and T<sub>B</sub>, to safely exit.

**FIGURE 40-4: TYPICAL LOW-VOLTAGE DETECT APPLICATION**



## 40.6 Operation During Sleep

When enabled, the HLVD circuitry continues to operate during Sleep. If the device voltage crosses the trip point, the HLVDIF bit will be set and the device will wake up from Sleep. Device execution will continue from the interrupt vector address if interrupts have been globally enabled.

## 40.7 Operation During Idle and Doze Modes

In both Idle and Doze modes, the module is active and events are generated if peripheral is enabled.

## 40.8 Operation During Freeze

When in Freeze mode, no new event or interrupt can be generated. The state of the LRDY bit is frozen.

Register reads and writes through the CPU interface are allowed.

## 40.9 Effects of a Reset

A device Reset forces all registers to their Reset state. This forces the HLVD module to be turned off.

## 40.10 Register Definitions: HLVD Control

Long bit name prefixes for the HLVD peripheral is shown in [Table 40-1](#). Refer to [Section 1.3.2.2 “Long Bit Names”](#) for more information.

**TABLE 40-1:**

| Peripheral | Bit Name Prefix |
|------------|-----------------|
| HLVD       | HLVD            |

**REGISTER 40-1: HLVDCON0: HIGH/LOW-VOLTAGE DETECT CONTROL REGISTER 0**

|         |     |     |     |     |     |         |         |
|---------|-----|-----|-----|-----|-----|---------|---------|
| R/W-0/0 | U-0 | R-x | R-x | U-0 | U-0 | R/W-0/0 | R/W-0/0 |
| EN      | —   | OUT | RDY | —   | —   | INTH    | INTL    |
| bit 7   |     |     |     |     |     |         | bit 0   |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as ‘0’

-n = Value at POR

‘1’ = Bit is set

‘0’ = Bit is cleared

x = Bit is unknown

- bit 7 **EN:** High/Low-voltage Detect Power Enable bit  
 1 = Enables HLVD, powers up HLVD circuit and supporting reference circuitry  
 0 = Disables HLVD, powers down HLVD and supporting circuitry
- bit 6 **Unimplemented:** Read as ‘0’
- bit 5 **OUT:** HLVD Comparator Output bit  
 1 = Voltage  $\leq$  selected detection limit (HLVDL<3:0>)  
 0 = Voltage  $\geq$  selected detection limit (HLVDL<3:0>)
- bit 4 **RDY:** Band Gap Reference Voltages Stable Status Flag bit  
 1 = Indicates HLVD Module is ready and output is stable  
 0 = Indicates HLVD Module is not ready
- bit 3-2 **Unimplemented:** Read as ‘0’
- bit 1 **INTH:** HLVD Positive going (High Voltage) Interrupt Enable  
 1 = HLVDIF will be set when voltage  $\geq$  selected detection limit (SEL<3:0>)  
 0 = HLVDIF will not be set
- bit 0 **INTL:** HLVD Negative going (Low Voltage) Interrupt Enable  
 1 = HLVDIF will be set when voltage  $\leq$  selected detection limit (SEL<3:0>)  
 0 = HLVDIF will not be set

**REGISTER 40-2: HLVDCON1: LOW-VOLTAGE DETECT CONTROL REGISTER 1**

|       |     |     |     |          |         |         |         |
|-------|-----|-----|-----|----------|---------|---------|---------|
| U-0   | U-0 | U-0 | U-0 | R/W-0/u  | R/W-0/u | R/W-0/u | R/W-0/u |
| —     | —   | —   | —   | SEL<3:0> |         |         |         |
| bit 7 |     |     |     | bit 0    |         |         |         |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

u = Bit is unchanged

bit 7-4

**Unimplemented:** Read as '0'

bit 3-0

**SEL<3:0>:** High/Low Voltage Detection Limit Selection bits

| SEL<3:0> | Typical Voltage |
|----------|-----------------|
| 1111     | Reserved        |
| 1110     | 4.65V           |
| 1101     | 4.35V           |
| 1100     | 4.20V           |
| 1011     | 4.00V           |
| 1010     | 3.75V           |
| 1001     | 3.60V           |
| 1000     | 3.35V           |
| 0111     | 3.15V           |
| 0110     | 2.90V           |
| 0101     | 2.75V           |
| 0100     | 2.60V           |
| 0011     | 2.50V           |
| 0010     | 2.25V           |
| 0001     | 2.10V           |
| 0000     | 1.90V           |

**TABLE 40-2: SUMMARY OF REGISTERS ASSOCIATED WITH HIGH/LOW-VOLTAGE DETECT MODULE**

| Name     | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3    | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|----------|-------|-------|-------|-------|----------|-------|-------|-------|------------------|
| HLVDCON0 | EN    | —     | OUT   | RDY   | —        | —     | INTH  | INTL  | 711              |
| HLVDCON1 | —     | —     | —     | —     | SEL<3:0> |       |       |       | 712              |

**Legend:** — = unimplemented, read as '0'. Shaded cells are unused by the HLVD module.



## 41.0 IN-CIRCUIT SERIAL PROGRAMMING™ (ICSP™)

ICSP™ programming allows customers to manufacture circuit boards with unprogrammed devices. Programming can be done after the assembly process, allowing the device to be programmed with the most recent firmware or a custom firmware. Five pins are needed for ICSP™ programming:

- ICSPCLK
- ICSPDAT
- MCLR/VPP
- VDD
- VSS

In Program/Verify mode the program memory, User IDs and the Configuration Words are programmed through serial communications. The ICSPDAT pin is a bidirectional I/O used for transferring the serial data and the ICSPCLK pin is the clock input. For more information on ICSP™ refer to the “*PIC18(L)F25/26K82 Memory Programming Specification*” (DS40000000).

### 41.1 High-Voltage Programming Entry Mode

The device is placed into High-Voltage Programming Entry mode by holding the ICSPCLK and ICSPDAT pins low then raising the voltage on MCLR/VPP to VIH.

### 41.2 Low-Voltage Programming Entry Mode

The Low-Voltage Programming Entry mode allows the PIC® Flash MCUs to be programmed using VDD only, without high voltage. When the LVP bit of Configuration Words is set to ‘1’, the low-voltage ICSP™ programming entry is enabled. To disable the Low-Voltage ICSP mode, the LVP bit must be programmed to ‘0’.

Entry into the Low-Voltage Programming Entry mode requires the following steps:

1.  $\overline{\text{MCLR}}$  is brought to VIL.
2. A 32-bit key sequence is presented on ICSPDAT, while clocking ICSPCLK.

Once the key sequence is complete,  $\overline{\text{MCLR}}$  must be held at VIL for as long as Program/Verify mode is to be maintained.

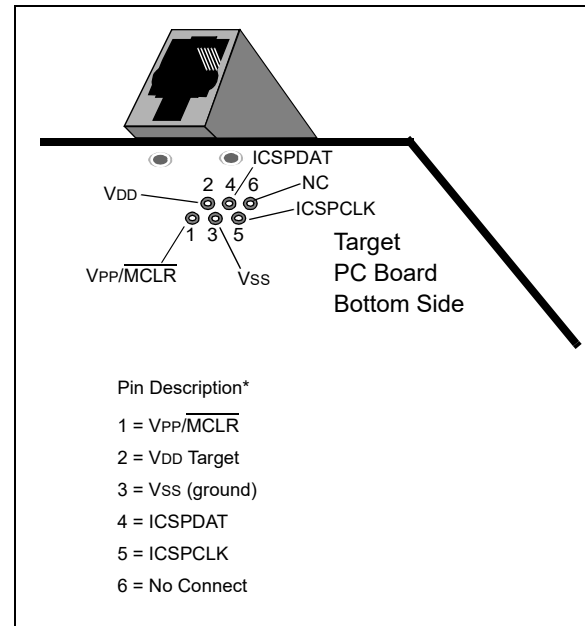
If low-voltage programming is enabled (LVP = 1), the MCLR Reset function is automatically enabled and cannot be disabled. See [Section 6.5 “MCLR”](#) for more information.

The LVP bit can only be reprogrammed to ‘0’ by using the High-Voltage Programming mode.

## 41.3 Common Programming Interfaces

Connection to a target device is typically done through an ICSP™ header. A commonly found connector on development tools is the RJ-11 in the 6P6C (6-pin, 6-conductor) configuration. See [Figure 41-1](#).

**FIGURE 41-1: ICD RJ-11 STYLE CONNECTOR INTERFACE**



Another connector often found in use with the PICkit™ programmers is a standard 6-pin header with 0.1 inch spacing. Refer to [Figure 41-2](#).

For additional interface recommendations, refer to your specific device programmer manual prior to PCB design.

It is recommended that isolation devices be used to separate the programming pins from other circuitry. The type of isolation is highly dependent on the specific application and may include devices such as resistors, diodes, or even jumpers. See [Figure 41-3](#) for more information.

**FIGURE 41-2: PICKIT™ PROGRAMMER STYLE CONNECTOR INTERFACE**

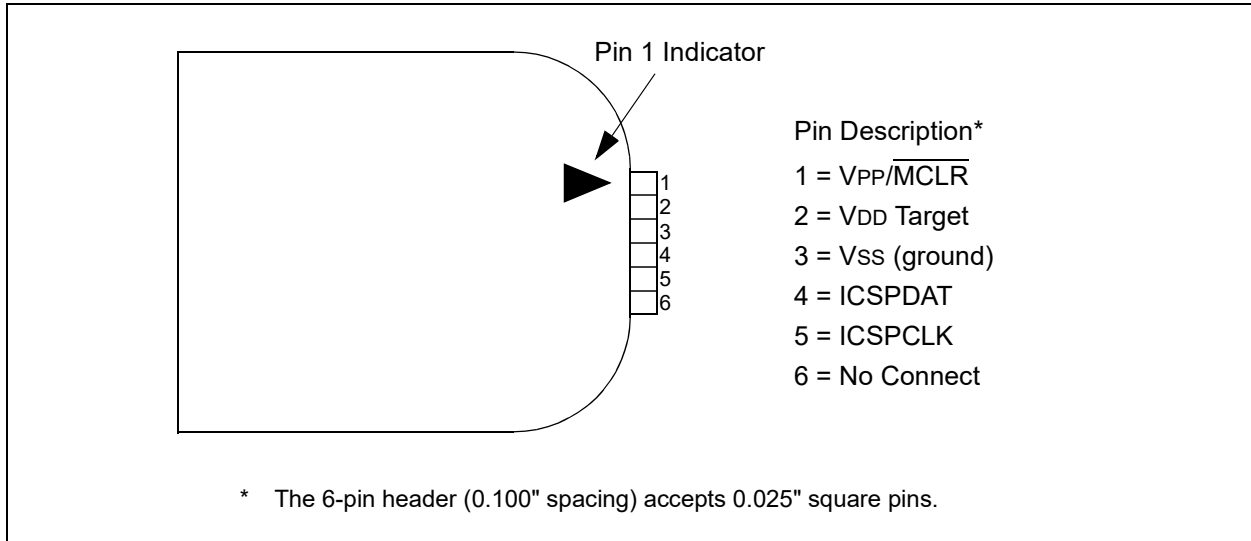
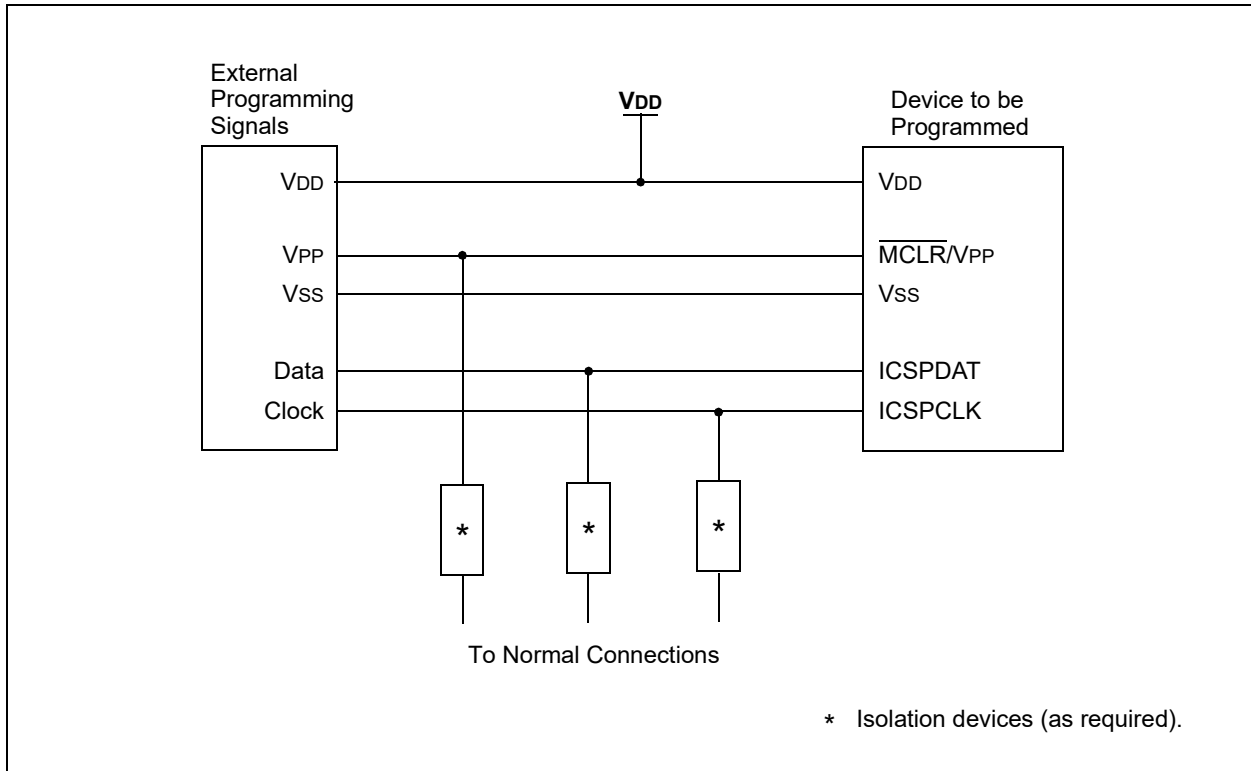


**FIGURE 41-3: TYPICAL CONNECTION FOR ICSP™ PROGRAMMING**



## 42.0 INSTRUCTION SET SUMMARY

PIC18(L)F25/26K83 devices incorporate the standard set of PIC18 core instructions, as well as an extended set of instructions, for the optimization of code that is recursive or that utilizes a software stack. The extended set is discussed later in this section.

### 42.1 Standard Instruction Set

The standard PIC18 instruction set adds many enhancements to the previous PIC<sup>®</sup> MCU instruction sets, while maintaining an easy migration from these PIC<sup>®</sup> MCU instruction sets. Most instructions are a single program memory word (16 bits), but there are four instructions that require two-program memory locations and two that require three-program memory locations.

Each single-word instruction is a 16-bit word divided into an opcode, which specifies the instruction type and one or more operands, which further specify the operation of the instruction.

The instruction set is highly orthogonal and is grouped into four basic categories:

- **Byte-oriented** operations
- **Bit-oriented** operations
- **Literal** operations
- **Control** operations

The PIC18 instruction set summary in [Table 42-3](#) lists **byte-oriented**, **bit-oriented**, **literal** and **control** operations. [Table 42-1](#) shows the opcode field descriptions.

Most **byte-oriented** instructions have three operands:

1. The file register (specified by 'f')
2. The destination of the result (specified by 'd')
3. The accessed memory (specified by 'a')

The file register designator 'f' specifies which file register is to be used by the instruction. The destination designator 'd' specifies where the result of the operation is to be placed. If 'd' is zero, the result is placed in the WREG register. If 'd' is one, the result is placed in the file register specified in the instruction.

All **bit-oriented** instructions have three operands:

1. The file register (specified by 'f')
2. The bit in the file register (specified by 'b')
3. The accessed memory (specified by 'a')

The bit field designator 'b' selects the number of the bit affected by the operation, while the file register designator 'f' represents the number of the file in which the bit is located.

The literal instructions may use some of the following operands:

- A literal value to be loaded into a file register (specified by 'k')
- The desired FSR register to load the literal value into (specified by 'f')
- No operand required (specified by '—')

The control instructions may use some of the following operands:

- A program memory address (specified by 'n')
- The mode of the CALL or RETURN instructions (specified by 's')
- The mode of the table read and table write instructions (specified by 'm')
- No operand required (specified by '—')

All instructions are a single word, except for four double-word instructions. These instructions were made double-word to contain the required information in 32 bits. In the second word, the four MSBs are '1's. If this second word is executed as an instruction (by itself), it will execute as a NOP.

All single-word instructions are executed in a single instruction cycle, unless a conditional test is true or the Program Counter is changed as a result of the instruction. In these cases, the execution takes two instruction cycles, with the additional instruction cycle(s) executed as a NOP.

The double-word instructions execute in two instruction cycles.

One instruction cycle consists of four oscillator periods. Thus, for an oscillator frequency of 4 MHz, the normal instruction execution time is 1  $\mu$ s. If a conditional test is true, or the Program Counter is changed as a result of an instruction, the instruction execution time is 2  $\mu$ s. Two-word branch instructions (if true) would take 3  $\mu$ s.

[Figure 42-1](#) shows the general formats that the instructions can have. All examples use the convention 'nnh' to represent a hexadecimal number.

The Instruction Set Summary, shown in [Table 42-3](#), lists the standard instructions recognized by the Microchip Assembler (MPASM<sup>™</sup>).

[Section 42.1.1 "Standard Instruction Set"](#) provides a description of each instruction.

**TABLE 42-1: OPCODE FIELD DESCRIPTIONS**

| Field           | Description                                                                                                                                                      |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| a               | RAM access bit<br>a = 0: RAM location in Access RAM (BSR register is ignored)<br>a = 1: RAM bank is specified by BSR register                                    |
| ACCESS          | ACCESS = 0: RAM access bit symbol                                                                                                                                |
| BANKED          | BANKED = 1: RAM access bit symbol                                                                                                                                |
| bbb             | Bit address within an 8-bit file register (0 to 7)                                                                                                               |
| BSR             | Bank Select Register. Used to select the current RAM bank.                                                                                                       |
| d               | Destination select bit;<br>d = 0: store result in WREG,<br>d = 1: store result in file register f.                                                               |
| dest            | Destination either the WREG register or the specified register file location                                                                                     |
| f               | 8-bit Register file address (00h to FFh)                                                                                                                         |
| f <sub>n</sub>  | FSR Number (0 to 2)                                                                                                                                              |
| f <sub>s</sub>  | 12-bit Register file address (000h to FFFh). This is the source address.                                                                                         |
| f <sub>d</sub>  | 12-bit Register file address (000h to FFFh). This is the destination address.                                                                                    |
| z <sub>s</sub>  | 7-bit literal offset for FSR2 to used as register file address (000h to FFFh). This is the source address.                                                       |
| z <sub>d</sub>  | 7-bit literal offset for FSR2 to used as register file address (000h to FFFh). This is the destination address.                                                  |
| k               | Literal field, constant data or label (may be a 6-bit, 8-bit, 12-bit or a 20-bit value)                                                                          |
| label           | Label name                                                                                                                                                       |
| mm              | The mode of the TBLPTR register for the Table Read and Table Write instructions<br>Only used with Table Read and Table Write instructions:                       |
| *               | No Change to register (such as TBLPTR with Table reads and writes)                                                                                               |
| *+              | Post-Increment register (such as TBLPTR with Table reads and writes)                                                                                             |
| *-              | Post-Decrement register (such as TBLPTR with Table reads and writes)                                                                                             |
| ++              | Pre-Increment register (such as TBLPTR with Table reads and writes)                                                                                              |
| n               | The relative address (2's complement number) for relative branch instructions, or the direct address for Call/Branch and Return instructions                     |
| PRODH           | Product of Multiply high byte                                                                                                                                    |
| PRODL           | Product of Multiply low byte                                                                                                                                     |
| s               | Fast Call / Return mode select bit.<br>s = 0: do not update into/from shadow registers<br>s = 1: certain registers loaded into/from shadow registers (Fast mode) |
| u               | Unused or Unchanged                                                                                                                                              |
| W               | W = 0: Destination select bit symbol                                                                                                                             |
| WREG            | Working register (accumulator)                                                                                                                                   |
| x               | Don't care (0 or 1)<br>The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.       |
| TBLPTR          | 21-bit Table Pointer (points to a Program Memory location)                                                                                                       |
| TABLAT          | 8-bit Table Latch                                                                                                                                                |
| TOS             | Top-of-Stack                                                                                                                                                     |
| PC              | Program Counter                                                                                                                                                  |
| PCL             | Program Counter Low Byte                                                                                                                                         |
| PCH             | Program Counter High Byte                                                                                                                                        |
| PCLATH          | Program Counter High Byte Latch                                                                                                                                  |
| PCLATU          | Program Counter Upper Byte Latch                                                                                                                                 |
| GIE             | Global Interrupt Enable bit                                                                                                                                      |
| WDT             | Watchdog Timer                                                                                                                                                   |
| TO              | Time-out bit                                                                                                                                                     |
| PD              | Power-down bit                                                                                                                                                   |
| C, DC, Z, OV, N | ALU Status bits Carry, Digit Carry, Zero, Overflow, Negative                                                                                                     |
| [ ]             | Optional                                                                                                                                                         |
| ( )             | Contents                                                                                                                                                         |
| →               | Assigned to                                                                                                                                                      |

**TABLE 42-1: OPCODE FIELD DESCRIPTIONS (CONTINUED)**

| Field          | Description                         |
|----------------|-------------------------------------|
| < >            | Register bit field                  |
| ∈              | In the set of                       |
| <i>italics</i> | User defined term (font is courier) |

**FIGURE 42-1: General Format for Instructions (1/2)**

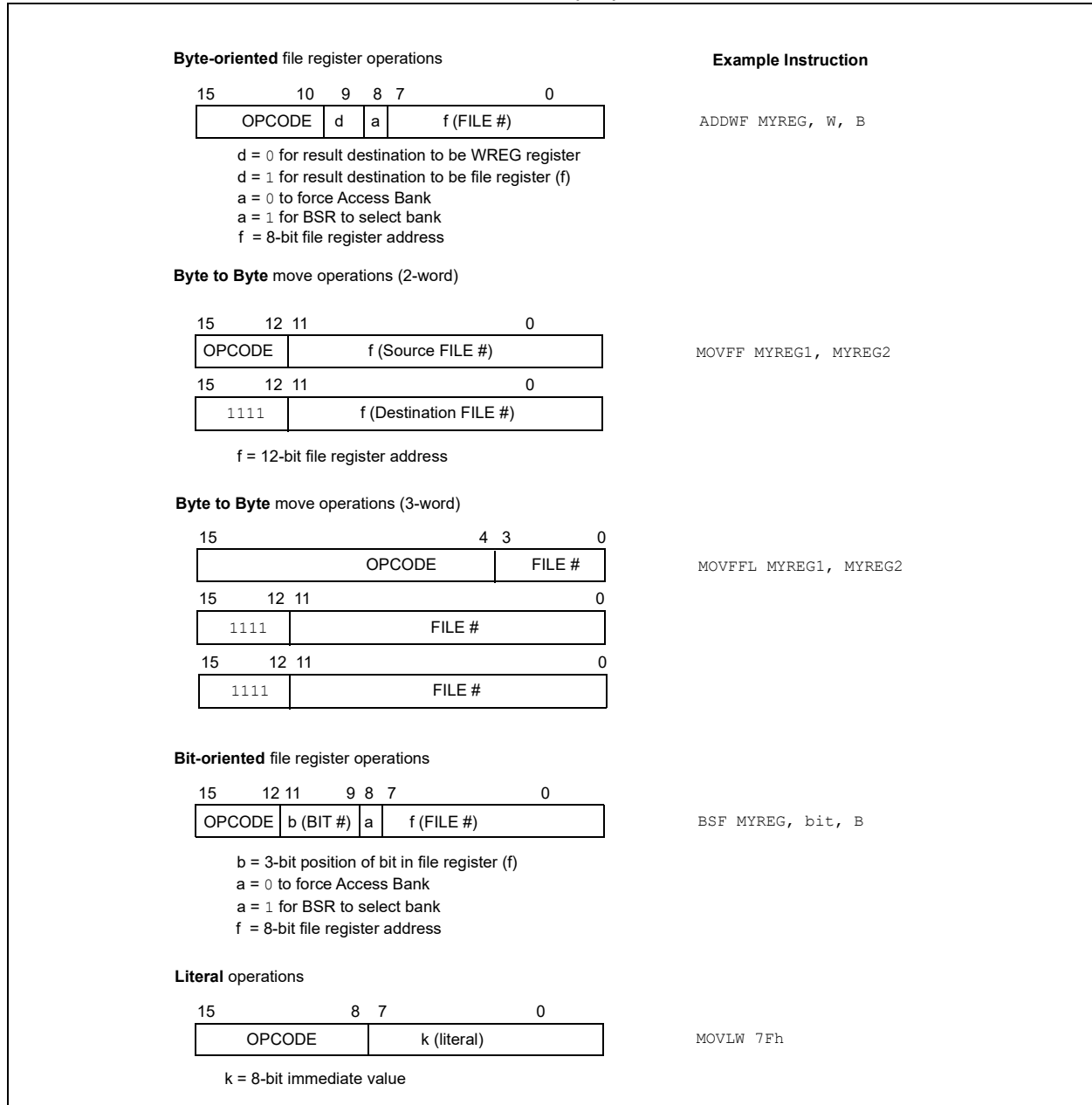
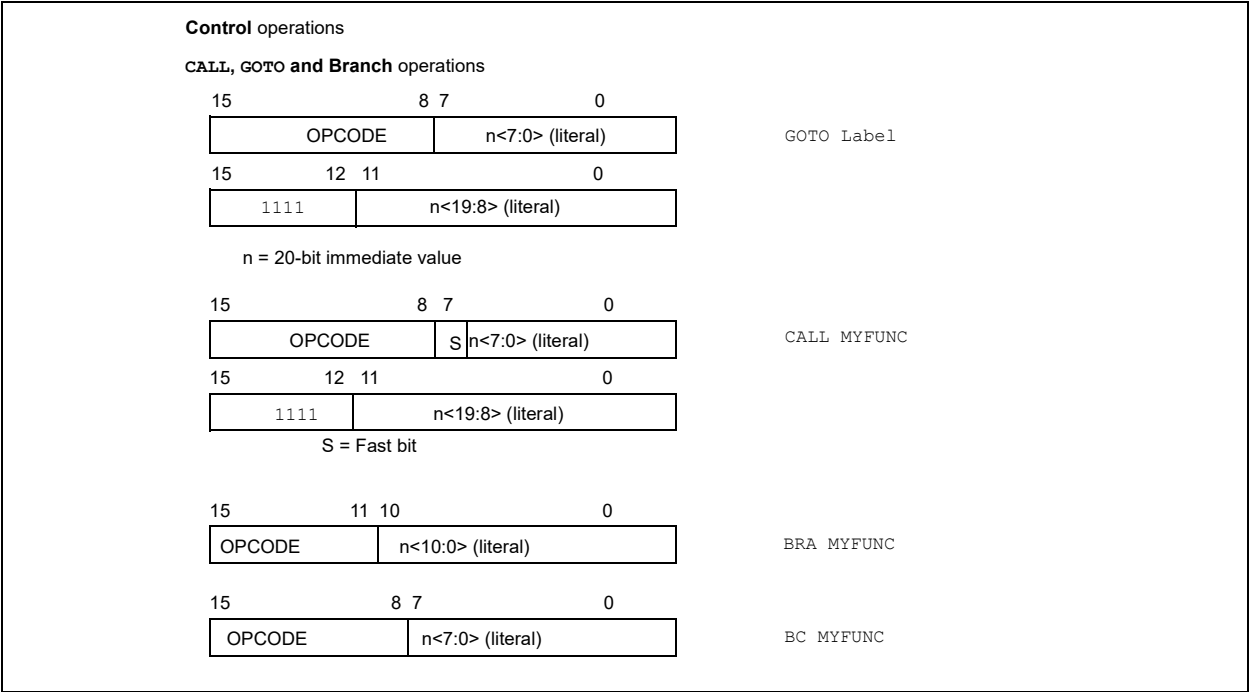


FIGURE 42-2: General Format for Instructions (2/2)



# PIC18(L)F25/26K83

**TABLE 42-2: INSTRUCTION SET**

| Mnemonic,<br>Operands                    |                                 | Description                                                                                          | Cycles     | 16-Bit Instruction Word |      |      |      | Status<br>Affected | Notes |
|------------------------------------------|---------------------------------|------------------------------------------------------------------------------------------------------|------------|-------------------------|------|------|------|--------------------|-------|
|                                          |                                 |                                                                                                      |            | MSb                     |      | LSb  |      |                    |       |
| BYTE-ORIENTED FILE REGISTER INSTRUCTIONS |                                 |                                                                                                      |            |                         |      |      |      |                    |       |
| ADDWF                                    | f, d ,a                         | Add WREG and f                                                                                       | 1          | 0010                    | 01da | ffff | ffff | C, DC, Z, OV, N    |       |
| ADDWFC                                   | f, d, a                         | Add WREG and Carry bit to f                                                                          | 1          | 0010                    | 00da | ffff | ffff | C, DC, Z, OV, N    |       |
| ANDWF                                    | f, d, a                         | AND WREG with f                                                                                      | 1          | 0001                    | 01da | ffff | ffff | Z, N               |       |
| CLRF                                     | f, a                            | Clear f                                                                                              | 1          | 0110                    | 101a | ffff | ffff | Z                  |       |
| COMF                                     | f, d, a                         | Complement f                                                                                         | 1          | 0001                    | 11da | ffff | ffff | Z, N               |       |
| DECf                                     | f, d, a                         | Decrement f                                                                                          | 1          | 0000                    | 01da | ffff | ffff | C, DC, Z, OV, N    |       |
| INCF                                     | f, d, a                         | Increment f                                                                                          | 1          | 0010                    | 10da | ffff | ffff | C, DC, Z, OV, N    |       |
| IORWF                                    | f, d, a                         | Inclusive OR WREG with f                                                                             | 1          | 0001                    | 00da | ffff | ffff | Z, N               |       |
| MOVF                                     | f, d, a                         | Move f to WREG or f                                                                                  | 1          | 0101                    | 00da | ffff | ffff | Z, N               |       |
| MOVFF                                    | f <sub>s</sub> , f <sub>d</sub> | Move f <sub>s</sub> (source) to 1st word<br>f <sub>d</sub> (destination) 2nd word                    | 2          | 1100                    | ffff | ffff | ffff | None               |       |
| MOVFFL                                   | f <sub>s</sub> , f <sub>d</sub> | Move f <sub>s</sub> (source) to<br>g (full destination)<br>f <sub>d</sub> (full destination)3rd word | 3          | 0000                    | 0000 | 0110 | ffff | None               | 2     |
|                                          |                                 |                                                                                                      |            | 1111                    | ffff | ffff | ffgg |                    |       |
|                                          |                                 |                                                                                                      |            | 1111                    | gggg | gggg | gggg |                    |       |
|                                          |                                 |                                                                                                      |            | 0110                    | 111a | ffff | ffff |                    |       |
| MOVWF                                    | f, a                            | Move WREG to f                                                                                       | 1          | 0110                    | 111a | ffff | ffff | None               |       |
| MULWF                                    | f, a                            | Multiply WREG with f                                                                                 | 1          | 0000                    | 001a | ffff | ffff | None               |       |
| NEGF                                     | f, a                            | Negate f                                                                                             | 1          | 0110                    | 110a | ffff | ffff | C, DC, Z, OV, N    |       |
| RLCF                                     | f, d, a                         | Rotate Left f through Carry                                                                          | 1          | 0011                    | 01da | ffff | ffff | C, Z, N            |       |
| RLNCF                                    | f, d, a                         | Rotate Left f (No Carry)                                                                             | 1          | 0100                    | 01da | ffff | ffff | Z, N               |       |
| RRCF                                     | f, d, a                         | Rotate Right f through Carry                                                                         | 1          | 0011                    | 00da | ffff | ffff | C, Z, N            |       |
| RRNCF                                    | f, d, a                         | Rotate Right f (No Carry)                                                                            | 1          | 0100                    | 00da | ffff | ffff | Z, N               |       |
| SETF                                     | f, a                            | Set f                                                                                                | 1          | 0110                    | 100a | ffff | ffff | None               |       |
| SUBFWB                                   | f, d, a                         | Subtract f from WREG with<br>borrow                                                                  | 1          | 0101                    | 01da | ffff | ffff | C, DC, Z, OV, N    |       |
| SUBWF                                    | f, d, a                         | Subtract WREG from f                                                                                 | 1          | 0101                    | 11da | ffff | ffff | C, DC, Z, OV, N    |       |
| SUBWFB                                   | f, d, a                         | Subtract WREG from f with<br>borrow                                                                  | 1          | 0101                    | 10da | ffff | ffff | C, DC, Z, OV, N    |       |
| SWAPF                                    | f, d, a                         | Swap nibbles in f                                                                                    | 1          | 0011                    | 10da | ffff | ffff | None               |       |
| XORWF                                    | f, d, a                         | Exclusive OR WREG with f                                                                             | 1          | 0001                    | 10da | ffff | ffff | Z, N               |       |
| BYTE-ORIENTED SKIP INSTRUCTIONS          |                                 |                                                                                                      |            |                         |      |      |      |                    |       |
| CPFSEQ                                   | f, a                            | Compare f with WREG, skip =                                                                          | 1 (2 or 3) | 0110                    | 001a | ffff | ffff | None               | 1     |
| CPFSGT                                   | f, a                            | Compare f with WREG, skip >                                                                          | 1 (2 or 3) | 0110                    | 010a | ffff | ffff | None               | 1     |
| CPFSLT                                   | f, a                            | Compare f with WREG, skip <                                                                          | 1 (2 or 3) | 0110                    | 000a | ffff | ffff | None               | 1     |
| DECFSZ                                   | f, d, a                         | Decrement f, Skip if 0                                                                               | 1 (2 or 3) | 0010                    | 11da | ffff | ffff | None               | 1     |
| DCFSNZ                                   | f, d, a                         | Decrement f, Skip if Not 0                                                                           | 1 (2 or 3) | 0100                    | 11da | ffff | ffff | None               | 1     |
| INCFSZ                                   | f, d, a                         | Increment f, Skip if 0                                                                               | 1 (2 or 3) | 0011                    | 11da | ffff | ffff | None               | 1     |
| INFSNZ                                   | f, d, a                         | Increment f, Skip if Not 0                                                                           | 1 (2 or 3) | 0100                    | 10da | ffff | ffff | None               | 1     |
| TSTFSZ                                   | f, a                            | Test f, skip if 0                                                                                    | 1 (2 or 3) | 0110                    | 011a | ffff | ffff | None               | 1     |
| BIT-ORIENTED FILE REGISTER INSTRUCTIONS  |                                 |                                                                                                      |            |                         |      |      |      |                    |       |
| BCF                                      | f, b, a                         | Bit Clear f                                                                                          | 1          | 1001                    | bbba | ffff | ffff | None               |       |
| BSF                                      | f, b, a                         | Bit Set f                                                                                            | 1          | 1000                    | bbba | ffff | ffff | None               |       |
| BTG                                      | f, d, a                         | Bit Toggle f                                                                                         | 1          | 0111                    | bbba | ffff | ffff | None               |       |
| BIT-ORIENTED SKIP INSTRUCTIONS           |                                 |                                                                                                      |            |                         |      |      |      |                    |       |
| BTFSC                                    | f, b, a                         | Bit Test f, Skip if Clear                                                                            | 1 (2 or 3) | 1011                    | bbba | ffff | ffff | None               | 1     |
| BTFSS                                    | f, b, a                         | Bit Test f, Skip if Set                                                                              | 1 (2 or 3) | 1010                    | bbba | ffff | ffff | None               | 1     |

- Note 1:** If Program Counter (PC) is modified or a conditional test is true, the instruction requires an additional cycle. The extra cycle is executed as a NOP.
- 2:** Some instructions are multi word instructions. The second/third words of these instructions will be decoded as a NOP, unless the first word of the instruction retrieves the information embedded in these 16-bits. This ensures that all program memory locations have a valid instruction.
- 3:** f<sub>s</sub> and f<sub>d</sub> do not cover the full memory range. 2 MSBs of bank selection are forced to 'b00 to limit the range of these instructions to lower 4k addressing space.

**TABLE 42-2: INSTRUCTION SET (CONTINUED)**

| Mnemonic,<br>Operands |      | Description                    | Cycles | 16-Bit Instruction Word |      |      |      | Status<br>Affected | Notes |
|-----------------------|------|--------------------------------|--------|-------------------------|------|------|------|--------------------|-------|
|                       |      |                                |        | MSb                     |      | LSb  |      |                    |       |
| CONTROL INSTRUCTIONS  |      |                                |        |                         |      |      |      |                    |       |
| BC                    | n    | Branch if Carry                | 1 (2)  | 1110                    | 0010 | nnnn | nnnn | None               | 1     |
| BN                    | n    | Branch if Negative             | 1 (2)  | 1110                    | 0110 | nnnn | nnnn | None               | 1     |
| BNC                   | n    | Branch if Not Carry            | 1 (2)  | 1110                    | 0011 | nnnn | nnnn | None               | 1     |
| BNN                   | n    | Branch if Not Negative         | 1 (2)  | 1110                    | 0111 | nnnn | nnnn | None               | 1     |
| BN OV                 | n    | Branch if Not Overflow         | 1 (2)  | 1110                    | 0101 | nnnn | nnnn | None               | 1     |
| BNZ                   | n    | Branch if Not Zero             | 2      | 1110                    | 0001 | nnnn | nnnn | None               | 1     |
| BOV                   | n    | Branch if Overflow             | 1 (2)  | 1110                    | 0100 | nnnn | nnnn | None               | 1     |
| BRA                   | n    | Branch Unconditionally         | 1 (2)  | 1101                    | 0nnn | nnnn | nnnn | None               |       |
| BZ                    | n    | Branch if Zero                 | 1 (2)  | 1110                    | 0000 | nnnn | nnnn | None               | 1     |
| CALL                  | n, s | Call subroutine                | 2      | 1110                    | 110s | nnnn | nnnn | None               | 2     |
|                       |      | 1st word                       |        |                         |      |      |      |                    |       |
|                       |      | 2nd word                       |        | 1111                    | nnnn | nnnn | nnnn |                    |       |
| GOTO                  | n    | Go to address                  | 2      | 1110                    | 1111 | nnnn | nnnn | None               | 2     |
|                       | —    | 1st word                       |        |                         |      |      |      |                    |       |
|                       |      | 2nd word                       |        | 1111                    | nnnn | nnnn | nnnn |                    |       |
| CALLW                 | —    | W -> PCL and Call subroutine   | 2      | 0000                    | 0000 | 0001 | 0100 | None               | 1     |
| RCALL                 | n    | Relative Call                  | 2      | 1101                    | 1nnn | nnnn | nnnn | None               | 1     |
| RETFIE                | s    | Return from interrupt enable   | 2      | 0000                    | 0000 | 0001 | 000s | None               | 1     |
| RETLW                 | k    | Return with literal in WREG    | 2      | 0000                    | 1100 | kkkk | kkkk | None               | 1     |
| RETURN                | s    | Return from Subroutine         | 2      | 0000                    | 0000 | 0001 | 001s | None               | 1     |
| INHERENT INSTRUCTIONS |      |                                |        |                         |      |      |      |                    |       |
| CLRWDT                | —    | Clear Watchdog Timer           | 1      | 0000                    | 0000 | 0000 | 0100 | None               | 2     |
| DAW                   | —    | Decimal Adjust WREG            | 1      | 0000                    | 0000 | 0000 | 0111 | C                  |       |
| NOP                   | —    | No Operation                   | 1      | 0000                    | 0000 | 0000 | 0000 | None               |       |
| NOP                   | —    | No Operation                   | 1      | 1111                    | xxxx | xxxx | xxxx | None               |       |
| POP                   | —    | Pop top of return stack (TOS)  | 1      | 0000                    | 0000 | 0000 | 0110 | None               |       |
| PUSH                  | —    | Push top of return stack (TOS) | 1      | 0000                    | 0000 | 0000 | 0101 | None               |       |
| RESET                 |      | Software device Reset          | 1      | 0000                    | 0000 | 1111 | 1111 | All                |       |
| SLEEP                 | —    | Go into Standby mode           | 1      | 0000                    | 0000 | 0000 | 0011 | None               |       |

**Note 1:** If Program Counter (PC) is modified or a conditional test is true, the instruction requires an additional cycle. The extra cycle is executed as a NOP.

- 2:** Some instructions are multi word instructions. The second/third words of these instructions will be decoded as a NOP, unless the first word of the instruction retrieves the information embedded in these 16-bits. This ensures that all program memory locations have a valid instruction.
- 3:**  $f_s$  and  $f_d$  do not cover the full memory range. 2 MSBs of bank selection are forced to 'b00 to limit the range of these instructions to lower 4k addressing space.



**TABLE 42-2: INSTRUCTION SET (CONTINUED)**

| Mnemonic,<br>Operands                     |                    | Description                                            | Cycles | 16-Bit Instruction Word |      |      |      | Status<br>Affected | Notes |
|-------------------------------------------|--------------------|--------------------------------------------------------|--------|-------------------------|------|------|------|--------------------|-------|
|                                           |                    |                                                        |        | MSb                     |      | LSb  |      |                    |       |
| LITERAL INSTRUCTIONS                      |                    |                                                        |        |                         |      |      |      |                    |       |
| ADDLW                                     | k                  | Add literal and WREG                                   | 1      | 0000                    | 1111 | kkkk | kkkk | C, DC, Z, OV, N    |       |
| ANDLW                                     | k                  | AND literal with WREG                                  | 1      | 0000                    | 1011 | kkkk | kkkk | Z, N               |       |
| IORLW                                     | k                  | Inclusive OR literal with WREG                         | 1      | 0000                    | 1001 | kkkk | kkkk | Z, N               |       |
| LFSR                                      | f <sub>n</sub> , k | Load FSR(f <sub>n</sub> ) with a 14-bit<br>literal (k) | 2      | 1110                    | 1110 | 00ff | kkkk | None               |       |
| ADDFSR                                    | f <sub>n</sub> , k | Add FSR(f <sub>n</sub> ) with (k)                      | 1      | 1110                    | 1000 | ffkk | kkkk | None               |       |
| SUBFSR                                    | f <sub>n</sub> , k | Subtract (k) from FSR(f <sub>n</sub> )                 | 1      | 1110                    | 1001 | ffkk | kkkk | None               |       |
| MOVLB                                     | k                  | Move literal to BSR<5:0>                               | 1      | 0000                    | 0001 | 00kk | kkkk | None               |       |
| MOVLW                                     | k                  | Move literal to WREG                                   | 1      | 0000                    | 1110 | kkkk | kkkk | None               |       |
| MULLW                                     | k                  | Multiply literal with WREG                             | 1      | 0000                    | 1101 | kkkk | kkkk | None               |       |
| RETLW                                     | k                  | Return with literal in WREG                            | 2      | 0000                    | 1100 | kkkk | kkkk | None               |       |
| SUBLW                                     | k                  | Subtract WREG from literal                             | 1      | 0000                    | 1000 | kkkk | kkkk | C, DC, Z, OV, N    |       |
| XORLW                                     | k                  | Exclusive OR literal with WREG                         | 1      | 0000                    | 1010 | kkkk | kkkk | Z, N               |       |
| DATA MEMORY – PROGRAM MEMORY INSTRUCTIONS |                    |                                                        |        |                         |      |      |      |                    |       |
| TBLRD*                                    |                    | Table Read                                             | 2 - 5  | 0000                    | 0000 | 0000 | 1000 | None               |       |
| TBLRD*+                                   |                    | Table Read with post-increment                         |        | 0000                    | 0000 | 0000 | 1001 | None               |       |
| TBLRD*-                                   |                    | Table Read with post-decrement                         | 0000   | 0000                    | 0000 | 1010 | None |                    |       |
| TBLRD+*                                   |                    | Table Read with pre-increment                          | 0000   | 0000                    | 0000 | 1011 | None |                    |       |
| TBLWT*                                    |                    | Table Write                                            | 2 - 5  | 0000                    | 0000 | 0000 | 1100 | None               |       |
| TBLWT*+                                   |                    | Table Write with post-increment                        |        | 0000                    | 0000 | 0000 | 1101 | None               |       |
| TBLWT*-                                   |                    | Table Write with post-decrement                        |        | 0000                    | 0000 | 0000 | 1110 | None               |       |
| TBLWT+*                                   |                    | Table Write with pre-increment                         |        | 0000                    | 0000 | 0000 | 1111 | None               |       |

- Note 1:** If Program Counter (PC) is modified or a conditional test is true, the instruction requires an additional cycle. The extra cycle is executed as a NOP.
- 2:** Some instructions are multi word instructions. The second/third words of these instructions will be decoded as a NOP, unless the first word of the instruction retrieves the information embedded in these 16-bits. This ensures that all program memory locations have a valid instruction.
- 3:** f<sub>s</sub> and f<sub>d</sub> do not cover the full memory range. 2 MSBs of bank selection are forced to 'b00 to limit the range of these instructions to lower 4k addressing space.

## 42.1.1 STANDARD INSTRUCTION SET

### ADDFSR Add Literal to FSR

Syntax: ADDFSR f, k

Operands:  $0 \leq k \leq 63$   
 $f \in [0, 1, 2]$

Operation:  $FSR(f) + k \rightarrow FSR(f)$

Status Affected: None

Encoding: 

|      |      |      |      |
|------|------|------|------|
| 1110 | 1000 | ffkk | kkkk |
|------|------|------|------|

Description: The 6-bit literal 'k' is added to the contents of the FSR specified by 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1     | Q2               | Q3           | Q4           |
|--------|------------------|--------------|--------------|
| Decode | Read literal 'k' | Process Data | Write to FSR |
| Decode | Read literal 'k' | Process Data | Write to FSR |

**Example:** ADDFSR 2, 23h

Before Instruction

FSR2 = 03FFh

After Instruction

FSR2 = 0422h

### ADDLW ADD literal to W

Syntax: ADDLW k

Operands:  $0 \leq k \leq 255$

Operation:  $(W) + k \rightarrow W$

Status Affected: N, OV, C, DC, Z

Encoding: 

|      |      |      |      |
|------|------|------|------|
| 0000 | 1111 | kkkk | kkkk |
|------|------|------|------|

Description: The contents of W are added to the 8-bit literal 'k' and the result is placed in W.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1     | Q2               | Q3           | Q4         |
|--------|------------------|--------------|------------|
| Decode | Read literal 'k' | Process Data | Write to W |

**Example:** ADDLW 15h

Before Instruction

W = 10h

After Instruction

W = 25h

### ADDWF ADD W to f

Syntax: ADDWF f {,d {,a}}

Operands:  $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

Operation:  $(W) + (f) \rightarrow \text{dest}$

Status Affected: N, OV, C, DC, Z

Encoding: 

|      |      |      |      |
|------|------|------|------|
| 0010 | 01da | ffff | ffff |
|------|------|------|------|

Description: Add W to register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default).

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh). See [Section 42.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"](#) for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1     | Q2                | Q3           | Q4                   |
|--------|-------------------|--------------|----------------------|
| Decode | Read register 'f' | Process Data | Write to destination |

**Example:** ADDWF REG, 0, 0

Before Instruction

W = 17h

REG = 0C2h

After Instruction

W = 0D9h

REG = 0C2h

**Note:** All PIC18 instructions may take an optional label argument preceding the instruction mnemonic for use in symbolic addressing. If a label is used, the instruction format then becomes: {label} instruction argument(s).

| ADDWFC            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | ADD W and CARRY bit to f |                      |  |  |  |      |      |      |      |        |                   |              |                      |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------|----------------------|--|--|--|------|------|------|------|--------|-------------------|--------------|----------------------|
| Syntax:           | ADDWFC    f {,d {,a}}                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                          |                      |  |  |  |      |      |      |      |        |                   |              |                      |
| Operands:         | $0 \leq f \leq 255$<br>$d \in [0,1]$<br>$a \in [0,1]$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                          |                      |  |  |  |      |      |      |      |        |                   |              |                      |
| Operation:        | $(W) + (f) + (C) \rightarrow \text{dest}$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                          |                      |  |  |  |      |      |      |      |        |                   |              |                      |
| Status Affected:  | N,OV, C, DC, Z                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                          |                      |  |  |  |      |      |      |      |        |                   |              |                      |
| Encoding:         | <table border="1"><tr><td>0010</td><td>00da</td><td>ffff</td><td>ffff</td></tr></table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                          |                      |  |  |  | 0010 | 00da | ffff | ffff |        |                   |              |                      |
| 0010              | 00da                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | ffff                     | ffff                 |  |  |  |      |      |      |      |        |                   |              |                      |
| Description:      | <p>Add W, the CARRY flag and data memory location 'f'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed in data memory location 'f'. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <a href="#">Section 42.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</a> for details.</p> |                          |                      |  |  |  |      |      |      |      |        |                   |              |                      |
| Words:            | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                          |                      |  |  |  |      |      |      |      |        |                   |              |                      |
| Cycles:           | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                          |                      |  |  |  |      |      |      |      |        |                   |              |                      |
| Q Cycle Activity: | <table><tr><td>Q1</td><td>Q2</td><td>Q3</td><td>Q4</td></tr><tr><td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr></table>                                                                                                                                                                                                                                                                                                                                                                                                           |                          |                      |  |  |  | Q1   | Q2   | Q3   | Q4   | Decode | Read register 'f' | Process Data | Write to destination |
| Q1                | Q2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | Q3                       | Q4                   |  |  |  |      |      |      |      |        |                   |              |                      |
| Decode            | Read register 'f'                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Process Data             | Write to destination |  |  |  |      |      |      |      |        |                   |              |                      |

**Example:** ADDWFC REG, 0, 1

Before Instruction

CARRY bit = 1  
 REG = 02h  
 W = 4Dh

After Instruction

CARRY bit = 0  
 REG = 02h  
 W = 50h

ANDLW

AND literal with W

Syntax:

ANDLW    k

Operands:

$0 \leq k \leq 255$

Operation:

$(W) .AND. k \rightarrow W$

Status Affected:

N, Z

Encoding:

|      |      |      |      |
|------|------|------|------|
| 0000 | 1011 | kkkk | kkkk |
|------|------|------|------|

Description:

The contents of W are AND'ed with the 8-bit literal 'k'. The result is placed in W.

Words:

1

Cycles:

1

Q Cycle Activity:

| Q1     | Q2               | Q3           | Q4         |
|--------|------------------|--------------|------------|
| Decode | Read literal 'k' | Process Data | Write to W |

**Example:** ANDLW 05Fh

Before Instruction

W = A3h

After Instruction

W = 03h

## ANDWF

## AND W with f

Syntax: ANDWF f {,d {,a}}

Operands:  $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

Operation: (W) .AND. (f) → dest

Status Affected: N, Z

Encoding: 

|      |      |      |      |
|------|------|------|------|
| 0001 | 01da | ffff | ffff |
|------|------|------|------|

Description: The contents of W are AND'ed with register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default). If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh). See [Section 42.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"](#) for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1     | Q2                | Q3           | Q4                   |
|--------|-------------------|--------------|----------------------|
| Decode | Read register 'f' | Process Data | Write to destination |

**Example:** ANDWF REG, 0, 0

Before Instruction

W = 17h  
 REG = C2h

After Instruction

W = 02h  
 REG = C2h

## BC

## Branch if Carry

Syntax: BC n

Operands:  $-128 \leq n \leq 127$

Operation: if CARRY bit is '1'  
 $(PC) + 2 + 2n \rightarrow PC$

Status Affected: None

Encoding: 

|      |      |      |      |
|------|------|------|------|
| 1110 | 0010 | nnnn | nnnn |
|------|------|------|------|

Description: If the CARRY bit is '1', then the program will branch.

The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be  $PC + 2 + 2n$ . This instruction is then a 2-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

If Jump:

| Q1           | Q2               | Q3           | Q4           |
|--------------|------------------|--------------|--------------|
| Decode       | Read literal 'n' | Process Data | Write to PC  |
| No operation | No operation     | No operation | No operation |

If No Jump:

| Q1     | Q2               | Q3           | Q4           |
|--------|------------------|--------------|--------------|
| Decode | Read literal 'n' | Process Data | No operation |

**Example:** HERE BC 5

Before Instruction

PC = address (HERE)

After Instruction

If CARRY = 1;  
 PC = address (HERE + 12)  
 If CARRY = 0;  
 PC = address (HERE + 2)

## BCF

## Bit Clear f

|                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                   |              |                    |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|--------------|--------------------|
| Syntax:           | BCF f, b {a}                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                   |              |                    |
| Operands:         | $0 \leq f \leq 255$<br>$0 \leq b \leq 7$<br>$a \in [0,1]$                                                                                                                                                                                                                                                                                                                                                                                                             |                   |              |                    |
| Operation:        | $0 \rightarrow f \leftarrow b$                                                                                                                                                                                                                                                                                                                                                                                                                                        |                   |              |                    |
| Status Affected:  | None                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                   |              |                    |
| Encoding:         | 1001                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | bbba              | ffff         | ffff               |
| Description:      | <p>Bit 'b' in register 'f' is cleared.</p> <p>If 'a' is '0', the Access Bank is selected.</p> <p>If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <a href="#">Section 42.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</a> for details.</p> |                   |              |                    |
| Words:            | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                   |              |                    |
| Cycles:           | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                   |              |                    |
| Q Cycle Activity: |                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                   |              |                    |
|                   | Q1                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Q2                | Q3           | Q4                 |
|                   | Decode                                                                                                                                                                                                                                                                                                                                                                                                                                                                | Read register 'f' | Process Data | Write register 'f' |

**Example:** BCF FLAG\_REG, 7, 0

Before Instruction  
 FLAG\_REG = C7h  
 After Instruction  
 FLAG\_REG = 47h

## BN

## Branch if Negative

|                   |                                                                                                                                                                                                                                                                                                |      |      |      |      |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|------|------|------|
| Syntax:           | BN    n                                                                                                                                                                                                                                                                                        |      |      |      |      |
| Operands:         | $-128 \leq n \leq 127$                                                                                                                                                                                                                                                                         |      |      |      |      |
| Operation:        | if NEGATIVE bit is '1'<br>$(PC) + 2 + 2n \rightarrow PC$                                                                                                                                                                                                                                       |      |      |      |      |
| Status Affected:  | None                                                                                                                                                                                                                                                                                           |      |      |      |      |
| Encoding:         | <table border="1"><tr><td>1110</td><td>0110</td><td>nnnn</td><td>nnnn</td></tr></table>                                                                                                                                                                                                        | 1110 | 0110 | nnnn | nnnn |
| 1110              | 0110                                                                                                                                                                                                                                                                                           | nnnn | nnnn |      |      |
| Description:      | <p>If the NEGATIVE bit is '1', then the program will branch.</p> <p>The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be <math>PC + 2 + 2n</math>. This instruction is then a 2-cycle instruction.</p> |      |      |      |      |
| Words:            | 1                                                                                                                                                                                                                                                                                              |      |      |      |      |
| Cycles:           | 1(2)                                                                                                                                                                                                                                                                                           |      |      |      |      |
| Q Cycle Activity: |                                                                                                                                                                                                                                                                                                |      |      |      |      |
| If Jump:          |                                                                                                                                                                                                                                                                                                |      |      |      |      |

| Q1           | Q2               | Q3           | Q4           |
|--------------|------------------|--------------|--------------|
| Decode       | Read literal 'n' | Process Data | Write to PC  |
| No operation | No operation     | No operation | No operation |

If No Jump:

| Q1     | Q2               | Q3           | Q4           |
|--------|------------------|--------------|--------------|
| Decode | Read literal 'n' | Process Data | No operation |

**Example:** HERE BN Jump

Before Instruction  
 PC = address (HERE)  
 After Instruction  
 If NEGATIVE = 1;  
 PC = address (Jump)  
 If NEGATIVE = 0;  
 PC = address (HERE + 2)

# PIC18(L)F25/26K83

## BNC

### Branch if Not Carry

|                   |                                                                                                                                                                                                                                                                                |      |      |      |      |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|------|------|------|
| Syntax:           | BNC    n                                                                                                                                                                                                                                                                       |      |      |      |      |
| Operands:         | $-128 \leq n \leq 127$                                                                                                                                                                                                                                                         |      |      |      |      |
| Operation:        | if CARRY bit is '0'<br>(PC) + 2 + 2n → PC                                                                                                                                                                                                                                      |      |      |      |      |
| Status Affected:  | None                                                                                                                                                                                                                                                                           |      |      |      |      |
| Encoding:         | <table border="1"><tr><td>1110</td><td>0011</td><td>nnnn</td><td>nnnn</td></tr></table>                                                                                                                                                                                        | 1110 | 0011 | nnnn | nnnn |
| 1110              | 0011                                                                                                                                                                                                                                                                           | nnnn | nnnn |      |      |
| Description:      | <p>If the CARRY bit is '0', then the program will branch.</p> <p>The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a 2-cycle instruction.</p> |      |      |      |      |
| Words:            | 1                                                                                                                                                                                                                                                                              |      |      |      |      |
| Cycles:           | 1(2)                                                                                                                                                                                                                                                                           |      |      |      |      |
| Q Cycle Activity: |                                                                                                                                                                                                                                                                                |      |      |      |      |
| If Jump:          |                                                                                                                                                                                                                                                                                |      |      |      |      |

| Q1           | Q2               | Q3           | Q4           |
|--------------|------------------|--------------|--------------|
| Decode       | Read literal 'n' | Process Data | Write to PC  |
| No operation | No operation     | No operation | No operation |

If No Jump:

| Q1     | Q2               | Q3           | Q4           |
|--------|------------------|--------------|--------------|
| Decode | Read literal 'n' | Process Data | No operation |

**Example:** HERE BNC Jump

Before Instruction  
PC = address (HERE)  
After Instruction  
If CARRY = 0;  
PC = address (Jump)  
If CARRY = 1;  
PC = address (HERE + 2)

## BNN

### Branch if Not Negative

|                   |                                                                                                                                                                                                                                                                                   |      |      |      |      |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|------|------|------|
| Syntax:           | BNN    n                                                                                                                                                                                                                                                                          |      |      |      |      |
| Operands:         | $-128 \leq n \leq 127$                                                                                                                                                                                                                                                            |      |      |      |      |
| Operation:        | if NEGATIVE bit is '0'<br>(PC) + 2 + 2n → PC                                                                                                                                                                                                                                      |      |      |      |      |
| Status Affected:  | None                                                                                                                                                                                                                                                                              |      |      |      |      |
| Encoding:         | <table border="1"><tr><td>1110</td><td>0111</td><td>nnnn</td><td>nnnn</td></tr></table>                                                                                                                                                                                           | 1110 | 0111 | nnnn | nnnn |
| 1110              | 0111                                                                                                                                                                                                                                                                              | nnnn | nnnn |      |      |
| Description:      | <p>If the NEGATIVE bit is '0', then the program will branch.</p> <p>The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a 2-cycle instruction.</p> |      |      |      |      |
| Words:            | 1                                                                                                                                                                                                                                                                                 |      |      |      |      |
| Cycles:           | 1(2)                                                                                                                                                                                                                                                                              |      |      |      |      |
| Q Cycle Activity: |                                                                                                                                                                                                                                                                                   |      |      |      |      |
| If Jump:          |                                                                                                                                                                                                                                                                                   |      |      |      |      |

| Q1           | Q2               | Q3           | Q4           |
|--------------|------------------|--------------|--------------|
| Decode       | Read literal 'n' | Process Data | Write to PC  |
| No operation | No operation     | No operation | No operation |

If No Jump:

| Q1     | Q2               | Q3           | Q4           |
|--------|------------------|--------------|--------------|
| Decode | Read literal 'n' | Process Data | No operation |

**Example:** HERE BNN Jump

Before Instruction  
PC = address (HERE)  
After Instruction  
If NEGATIVE = 0;  
PC = address (Jump)  
If NEGATIVE = 1;  
PC = address (HERE + 2)

# PIC18(L)F25/26K83

## BNOV Branch if Not Overflow

Syntax: BNOV n

Operands:  $-128 \leq n \leq 127$

Operation: if OVERFLOW bit is '0'  
 $(PC) + 2 + 2n \rightarrow PC$

Status Affected: None

Encoding: 

|      |      |      |      |
|------|------|------|------|
| 1110 | 0101 | nnnn | nnnn |
|------|------|------|------|

Description: If the OVERFLOW bit is '0', then the program will branch. The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be  $PC + 2 + 2n$ . This instruction is then a 2-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

If Jump:

| Q1           | Q2               | Q3           | Q4           |
|--------------|------------------|--------------|--------------|
| Decode       | Read literal 'n' | Process Data | Write to PC  |
| No operation | No operation     | No operation | No operation |

If No Jump:

| Q1     | Q2               | Q3           | Q4           |
|--------|------------------|--------------|--------------|
| Decode | Read literal 'n' | Process Data | No operation |

**Example:** HERE BNOV Jump

Before Instruction  
 PC = address (HERE)

After Instruction  
 If OVERFLOW = 0;  
 PC = address (Jump)  
 If OVERFLOW = 1;  
 PC = address (HERE + 2)

## BNZ Branch if Not Zero

Syntax: BNZ n

Operands:  $-128 \leq n \leq 127$

Operation: if ZERO bit is '0'  
 $(PC) + 2 + 2n \rightarrow PC$

Status Affected: None

Encoding: 

|      |      |      |      |
|------|------|------|------|
| 1110 | 0001 | nnnn | nnnn |
|------|------|------|------|

Description: If the ZERO bit is '0', then the program will branch. The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be  $PC + 2 + 2n$ . This instruction is then a 2-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

If Jump:

| Q1           | Q2               | Q3           | Q4           |
|--------------|------------------|--------------|--------------|
| Decode       | Read literal 'n' | Process Data | Write to PC  |
| No operation | No operation     | No operation | No operation |

If No Jump:

| Q1     | Q2               | Q3           | Q4           |
|--------|------------------|--------------|--------------|
| Decode | Read literal 'n' | Process Data | No operation |

**Example:** HERE BNZ Jump

Before Instruction  
 PC = address (HERE)

After Instruction  
 If ZERO = 0;  
 PC = address (Jump)  
 If ZERO = 1;  
 PC = address (HERE + 2)

## BRA Unconditional Branch

|                  |                                                                                                                                                                                                    |      |      |      |
|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|------|------|
| Syntax:          | BRA   n                                                                                                                                                                                            |      |      |      |
| Operands:        | $-1024 \leq n \leq 1023$                                                                                                                                                                           |      |      |      |
| Operation:       | $(PC) + 2 + 2n \rightarrow PC$                                                                                                                                                                     |      |      |      |
| Status Affected: | None                                                                                                                                                                                               |      |      |      |
| Encoding:        | 1101                                                                                                                                                                                               | 0nnn | nnnn | nnnn |
| Description:     | Add the 2's complement number '2n' to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be $PC + 2 + 2n$ . This instruction is a 2-cycle instruction. |      |      |      |
| Words:           | 1                                                                                                                                                                                                  |      |      |      |
| Cycles:          | 2                                                                                                                                                                                                  |      |      |      |

Q Cycle Activity:

| Q1           | Q2               | Q3           | Q4           |
|--------------|------------------|--------------|--------------|
| Decode       | Read literal 'n' | Process Data | Write to PC  |
| No operation | No operation     | No operation | No operation |

**Example:**                      HERE                      BRA    Jump

Before Instruction  
PC                      =    address    (HERE)

After Instruction  
PC                      =    address    (Jump)

## BSF Bit Set f

|                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                            |      |      |      |      |
|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|------|------|------|
| Syntax:          | BSF f, b {,a}                                                                                                                                                                                                                                                                                                                                                                                                                                              |      |      |      |      |
| Operands:        | $0 \leq f \leq 255$<br>$0 \leq b \leq 7$<br>$a \in [0,1]$                                                                                                                                                                                                                                                                                                                                                                                                  |      |      |      |      |
| Operation:       | $1 \rightarrow f \leftarrow b$                                                                                                                                                                                                                                                                                                                                                                                                                             |      |      |      |      |
| Status Affected: | None                                                                                                                                                                                                                                                                                                                                                                                                                                                       |      |      |      |      |
| Encoding:        | <table border="1"><tr><td>1000</td><td>bbba</td><td>ffff</td><td>ffff</td></tr></table>                                                                                                                                                                                                                                                                                                                                                                    | 1000 | bbba | ffff | ffff |
| 1000             | bbba                                                                                                                                                                                                                                                                                                                                                                                                                                                       | ffff | ffff |      |      |
| Description:     | <p>Bit 'b' in register 'f' is set.</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <a href="#">Section 42.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</a> for details.</p> |      |      |      |      |
| Words:           | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                          |      |      |      |      |
| Cycles:          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                          |      |      |      |      |

Q Cycle Activity:

| Q1     | Q2                | Q3           | Q4                 |
|--------|-------------------|--------------|--------------------|
| Decode | Read register 'f' | Process Data | Write register 'f' |

**Example:**                      BSF                      FLAG\_REG, 7, 1

Before Instruction  
FLAG\_REG                      =    0Ah

After Instruction  
FLAG\_REG                      =    8Ah



## BTFSC Bit Test File, Skip if Clear

Syntax: BTFSC f, b {,a}

Operands:  $0 \leq f \leq 255$   
 $0 \leq b \leq 7$   
 $a \in [0,1]$

Operation: skip if (f<b>) = 0

Status Affected: None

Encoding: 

|      |      |      |      |
|------|------|------|------|
| 1011 | bbba | ffff | ffff |
|------|------|------|------|

Description: If bit 'b' in register 'f' is '0', then the next instruction is skipped. If bit 'b' is '0', then the next instruction fetched during the current instruction execution is discarded and a NOP is executed instead, making this a 2-cycle instruction.  
 If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.  
 If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh).  
 See [Section 42.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"](#) for details.

Words: 1

Cycles: 1(2)  
**Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1     | Q2                | Q3           | Q4           |
|--------|-------------------|--------------|--------------|
| Decode | Read register 'f' | Process Data | No operation |

If skip:

| Q1           | Q2           | Q3           | Q4           |
|--------------|--------------|--------------|--------------|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1           | Q2           | Q3           | Q4           |
|--------------|--------------|--------------|--------------|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

**Example:**

|       |       |            |
|-------|-------|------------|
| HERE  | BTFSC | FLAG, 1, 0 |
| FALSE | :     |            |
| TRUE  | :     |            |

Before Instruction  
 PC = address (HERE)  
 After Instruction  
 If FLAG<1> = 0;  
     PC = address (TRUE)  
 If FLAG<1> = 1;  
     PC = address (FALSE)

## BTFSS Bit Test File, Skip if Set

Syntax: BTFSS f, b {,a}

Operands:  $0 \leq f \leq 255$   
 $0 \leq b < 7$   
 $a \in [0,1]$

Operation: skip if (f<b>) = 1

Status Affected: None

Encoding: 

|      |      |      |      |
|------|------|------|------|
| 1010 | bbba | ffff | ffff |
|------|------|------|------|

Description: If bit 'b' in register 'f' is '1', then the next instruction is skipped. If bit 'b' is '1', then the next instruction fetched during the current instruction execution is discarded and a NOP is executed instead, making this a 2-cycle instruction.  
 If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.  
 If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh).  
 See [Section 42.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"](#) for details.

Words: 1

Cycles: 1(2)  
**Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1     | Q2                | Q3           | Q4           |
|--------|-------------------|--------------|--------------|
| Decode | Read register 'f' | Process Data | No operation |

If skip:

| Q1           | Q2           | Q3           | Q4           |
|--------------|--------------|--------------|--------------|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1           | Q2           | Q3           | Q4           |
|--------------|--------------|--------------|--------------|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

**Example:**

|       |       |            |
|-------|-------|------------|
| HERE  | BTFSS | FLAG, 1, 0 |
| FALSE | :     |            |
| TRUE  | :     |            |

Before Instruction  
 PC = address (HERE)  
 After Instruction  
 If FLAG<1> = 0;  
     PC = address (FALSE)  
 If FLAG<1> = 1;  
     PC = address (TRUE)

## BTG

## Bit Toggle f

|                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                   |              |                    |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|--------------|--------------------|
| Syntax:           | BTG f, b {,a}                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                   |              |                    |
| Operands:         | $0 \leq f \leq 255$<br>$0 \leq b < 7$<br>$a \in [0,1]$                                                                                                                                                                                                                                                                                                                                                                                                                             |                   |              |                    |
| Operation:        | $(\overline{f < b >}) \rightarrow f < b >$                                                                                                                                                                                                                                                                                                                                                                                                                                         |                   |              |                    |
| Status Affected:  | None                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                   |              |                    |
| Encoding:         | 0111                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | bbba              | ffff         | ffff               |
| Description:      | <p>Bit 'b' in data memory location 'f' is inverted.</p> <p>If 'a' is '0', the Access Bank is selected.</p> <p>If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <a href="#">Section 42.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</a> for details.</p> |                   |              |                    |
| Words:            | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                   |              |                    |
| Cycles:           | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                   |              |                    |
| Q Cycle Activity: |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                   |              |                    |
|                   | Q1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | Q2                | Q3           | Q4                 |
|                   | Decode                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Read register 'f' | Process Data | Write register 'f' |

**Example:** BTG PORTC, 4, 0

Before Instruction:

PORTC = 0111 0101 [75h]

After Instruction:

PORTC = 0110 0101 [65h]

## BOV

## Branch if Overflow

|                   |                                                                                                                                                                                                                                                                                                |      |      |      |      |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|------|------|------|
| Syntax:           | BOV    n                                                                                                                                                                                                                                                                                       |      |      |      |      |
| Operands:         | $-128 \leq n \leq 127$                                                                                                                                                                                                                                                                         |      |      |      |      |
| Operation:        | if OVERFLOW bit is '1'<br>$(PC) + 2 + 2n \rightarrow PC$                                                                                                                                                                                                                                       |      |      |      |      |
| Status Affected:  | None                                                                                                                                                                                                                                                                                           |      |      |      |      |
| Encoding:         | <table border="1"><tr><td>1110</td><td>0100</td><td>nnnn</td><td>nnnn</td></tr></table>                                                                                                                                                                                                        | 1110 | 0100 | nnnn | nnnn |
| 1110              | 0100                                                                                                                                                                                                                                                                                           | nnnn | nnnn |      |      |
| Description:      | <p>If the OVERFLOW bit is '1', then the program will branch.</p> <p>The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be <math>PC + 2 + 2n</math>. This instruction is then a 2-cycle instruction.</p> |      |      |      |      |
| Words:            | 1                                                                                                                                                                                                                                                                                              |      |      |      |      |
| Cycles:           | 1(2)                                                                                                                                                                                                                                                                                           |      |      |      |      |
| Q Cycle Activity: |                                                                                                                                                                                                                                                                                                |      |      |      |      |
| If Jump:          |                                                                                                                                                                                                                                                                                                |      |      |      |      |

| Q1           | Q2               | Q3           | Q4           |
|--------------|------------------|--------------|--------------|
| Decode       | Read literal 'n' | Process Data | Write to PC  |
| No operation | No operation     | No operation | No operation |

If No Jump:

| Q1     | Q2               | Q3           | Q4           |
|--------|------------------|--------------|--------------|
| Decode | Read literal 'n' | Process Data | No operation |

**Example:** HERE BOV Jump

Before Instruction

PC = address (HERE)

After Instruction

If OVERFLOW = 1;

PC = address (Jump)

If OVERFLOW = 0;

PC = address (HERE + 2)

## BZ Branch if Zero

|                   |                                                                                                                                                                                                                                                                               |      |      |      |      |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|------|------|------|
| Syntax:           | BZ    n                                                                                                                                                                                                                                                                       |      |      |      |      |
| Operands:         | -128 ≤ n ≤ 127                                                                                                                                                                                                                                                                |      |      |      |      |
| Operation:        | if ZERO bit is '1'<br>(PC) + 2 + 2n → PC                                                                                                                                                                                                                                      |      |      |      |      |
| Status Affected:  | None                                                                                                                                                                                                                                                                          |      |      |      |      |
| Encoding:         | <table border="1"><tr><td>1110</td><td>0000</td><td>nnnn</td><td>nnnn</td></tr></table>                                                                                                                                                                                       | 1110 | 0000 | nnnn | nnnn |
| 1110              | 0000                                                                                                                                                                                                                                                                          | nnnn | nnnn |      |      |
| Description:      | <p>If the ZERO bit is '1', then the program will branch.</p> <p>The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a 2-cycle instruction.</p> |      |      |      |      |
| Words:            | 1                                                                                                                                                                                                                                                                             |      |      |      |      |
| Cycles:           | 1(2)                                                                                                                                                                                                                                                                          |      |      |      |      |
| Q Cycle Activity: |                                                                                                                                                                                                                                                                               |      |      |      |      |
| If Jump:          |                                                                                                                                                                                                                                                                               |      |      |      |      |

| Q1           | Q2               | Q3           | Q4           |
|--------------|------------------|--------------|--------------|
| Decode       | Read literal 'n' | Process Data | Write to PC  |
| No operation | No operation     | No operation | No operation |

If No Jump:

| Q1     | Q2               | Q3           | Q4           |
|--------|------------------|--------------|--------------|
| Decode | Read literal 'n' | Process Data | No operation |

**Example:**                    HERE                    BZ                    Jump

Before Instruction  
PC = address (HERE)  
After Instruction  
If ZERO = 1;  
PC = address (Jump)  
If ZERO = 0;  
PC = address (HERE + 2)

## CALL Subroutine Call

|                   |                                                                                                                                                              |                     |                    |                   |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------|--------------------|-------------------|
| Syntax:           | CALL k {,s}                                                                                                                                                  |                     |                    |                   |
| Operands:         | $0 \leq k \leq 1048575$<br>$s \in [0,1]$                                                                                                                     |                     |                    |                   |
| Operation:        | (PC) + 4 $\rightarrow$ TOS,<br>k $\rightarrow$ PC<20:1>,<br>if s = 1<br>(W) $\rightarrow$ WS,<br>(Status) $\rightarrow$ STATUSS,<br>(BSR) $\rightarrow$ BSRS |                     |                    |                   |
| Status Affected:  | None                                                                                                                                                         |                     |                    |                   |
| Encoding:         |                                                                                                                                                              |                     |                    |                   |
| 1st word (k<7:0>) | 1110                                                                                                                                                         | 110s                | k <sub>7</sub> kkk | kkkk <sub>0</sub> |
| 2nd word(k<19:8>) | 1111                                                                                                                                                         | k <sub>19</sub> kkk | kkkk               | kkkk <sub>8</sub> |
| Description:      | Subroutine call of entire 2-Mbyte                                                                                                                            |                     |                    |                   |

Words: 2

Cycles: 2

Q Cycle Activity:

| Q1           | Q2                     | Q3               | Q4                                  |
|--------------|------------------------|------------------|-------------------------------------|
| Decode       | Read literal 'k'<7:0>, | PUSH PC to stack | Read literal 'k'<19:8>, Write to PC |
| No operation | No operation           | No operation     | No operation                        |

**Example:**                    HERE                    CALL                    THERE, 1

Before Instruction  
PC = address (HERE)  
After Instruction  
PC = address (THERE)  
TOS = address (HERE + 4)  
WS = W  
BSRS = BSR  
STATUSS = Status

## CALLW Subroutine Call Using WREG

|                   |                                                                                                                                                                                                                                                                                                                                                                                                             |      |      |      |      |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|------|------|------|
| Syntax:           | CALLW                                                                                                                                                                                                                                                                                                                                                                                                       |      |      |      |      |
| Operands:         | None                                                                                                                                                                                                                                                                                                                                                                                                        |      |      |      |      |
| Operation:        | (PC + 2) → TOS,<br>(W) → PCL,<br>(PCLATH) → PCH,<br>(PCLATU) → PCU                                                                                                                                                                                                                                                                                                                                          |      |      |      |      |
| Status Affected:  | None                                                                                                                                                                                                                                                                                                                                                                                                        |      |      |      |      |
| Encoding:         | <table><tr><td>0000</td><td>0000</td><td>0001</td><td>0100</td></tr></table>                                                                                                                                                                                                                                                                                                                                | 0000 | 0000 | 0001 | 0100 |
| 0000              | 0000                                                                                                                                                                                                                                                                                                                                                                                                        | 0001 | 0100 |      |      |
| Description       | <p>First, the return address (PC + 2) is pushed onto the return stack. Next, the contents of W are written to PCL; the existing value is discarded. Then, the contents of PCLATH and PCLATU are latched into PCH and PCU, respectively. The second cycle is executed as a NOP instruction while the new next instruction is fetched.</p> <p>Unlike CALL, there is no option to update W, Status or BSR.</p> |      |      |      |      |
| Words:            | 1                                                                                                                                                                                                                                                                                                                                                                                                           |      |      |      |      |
| Cycles:           | 2                                                                                                                                                                                                                                                                                                                                                                                                           |      |      |      |      |
| Q Cycle Activity: |                                                                                                                                                                                                                                                                                                                                                                                                             |      |      |      |      |

| Q1           | Q2           | Q3               | Q4           |
|--------------|--------------|------------------|--------------|
| Decode       | Read WREG    | PUSH PC to stack | No operation |
| No operation | No operation | No operation     | No operation |

**Example:**            HERE      CALLW

Before Instruction

PC        =    address    (HERE)  
PCLATH =    10h  
PCLATU =    00h  
W        =    06h

After Instruction

PC        =    001006h  
TOS      =    address    (HERE + 2)  
PCLATH =    10h  
PCLATU =    00h  
W        =    06h

## CLRF Clear f

|                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |      |      |      |      |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|------|------|------|
| Syntax:           | CLRF   f {,a}                                                                                                                                                                                                                                                                                                                                                                                                                                                    |      |      |      |      |
| Operands:         | 0 ≤ f ≤ 255<br>a ∈ [0,1]                                                                                                                                                                                                                                                                                                                                                                                                                                         |      |      |      |      |
| Operation:        | 000h → f<br>1 → Z                                                                                                                                                                                                                                                                                                                                                                                                                                                |      |      |      |      |
| Status Affected:  | Z                                                                                                                                                                                                                                                                                                                                                                                                                                                                |      |      |      |      |
| Encoding:         | <table border="1"><tr><td>0110</td><td>101a</td><td>ffff</td><td>ffff</td></tr></table>                                                                                                                                                                                                                                                                                                                                                                          | 0110 | 101a | ffff | ffff |
| 0110              | 101a                                                                                                                                                                                                                                                                                                                                                                                                                                                             | ffff | ffff |      |      |
| Description:      | <p>Clears the contents of the specified register.</p> <p>If 'a' is '0', the Access Bank is selected.</p> <p>If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See <a href="#">Section 42.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</a> for details.</p> |      |      |      |      |
| Words:            | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                |      |      |      |      |
| Cycles:           | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                |      |      |      |      |
| Q Cycle Activity: |                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |      |      |      |      |

| Q1     | Q2                | Q3           | Q4                 |
|--------|-------------------|--------------|--------------------|
| Decode | Read register 'f' | Process Data | Write register 'f' |

**Example:**            CLRF            FLAG\_REG, 1

Before Instruction

FLAG\_REG =    5Ah

After Instruction

FLAG\_REG =    00h

## CLRWDT Clear Watchdog Timer

Syntax: CLRWDT

Operands: None

Operation: 000h → WDT,  
000h → WDT postscaler,  
1 →  $\overline{TO}$ ,  
1 → PD

Status Affected:  $\overline{TO}$ , PD

Encoding: 

|      |      |      |      |
|------|------|------|------|
| 0000 | 0000 | 0000 | 0100 |
|------|------|------|------|

Description: CLRWDT instruction resets the Watchdog Timer. It also resets the postscaler of the WDT. Status bits,  $\overline{TO}$  and PD, are set.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1     | Q2           | Q3           | Q4           |
|--------|--------------|--------------|--------------|
| Decode | No operation | Process Data | No operation |

**Example:** CLRWDT

Before Instruction  
WDT Counter = ?

After Instruction  
WDT Counter = 00h  
WDT Postscaler = 0  
 $\overline{TO}$  = 1  
PD = 1

## COMF Complement f

Syntax: COMF f {,d {,a}}

Operands:  $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

Operation:  $(\bar{f}) \rightarrow \text{dest}$

Status Affected: N, Z

Encoding: 

|      |      |      |      |
|------|------|------|------|
| 0001 | 11da | ffff | ffff |
|------|------|------|------|

Description: The contents of register 'f' are complemented. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default). If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh). See [Section 42.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"](#) for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1     | Q2                | Q3           | Q4                   |
|--------|-------------------|--------------|----------------------|
| Decode | Read register 'f' | Process Data | Write to destination |

**Example:** COMF REG, 0, 0

Before Instruction  
REG = 13h

After Instruction  
REG = 13h  
W = ECh

| CPFSEQ           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | Compare f with W, skip if f = W |      |  |  |      |      |      |      |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------|------|--|--|------|------|------|------|
| Syntax:          | CPFSEQ f {,a}                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                                 |      |  |  |      |      |      |      |
| Operands:        | $0 \leq f \leq 255$<br>$a \in [0,1]$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                                 |      |  |  |      |      |      |      |
| Operation:       | $(f) - (W)$ ,<br>skip if $(f) = (W)$<br>(unsigned comparison)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                                 |      |  |  |      |      |      |      |
| Status Affected: | None                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                                 |      |  |  |      |      |      |      |
| Encoding:        | <table border="1"><tr><td>0110</td><td>001a</td><td>ffff</td><td>ffff</td></tr></table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                                 |      |  |  | 0110 | 001a | ffff | ffff |
| 0110             | 001a                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | ffff                            | ffff |  |  |      |      |      |      |
| Description:     | <p>Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction. If 'f' = W, then the fetched instruction is discarded and a <code>NOP</code> is executed instead, making this a 2-cycle instruction.</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <a href="#">Section 42.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</a> for details.</p> |                                 |      |  |  |      |      |      |      |
| Words:           | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                                 |      |  |  |      |      |      |      |
| Cycles:          | 1(2)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                                 |      |  |  |      |      |      |      |
|                  | <b>Note:</b> 3 cycles if skip and followed by a 2-word instruction.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                                 |      |  |  |      |      |      |      |

Q Cycle Activity:

| Q1     | Q2                | Q3           | Q4           |
|--------|-------------------|--------------|--------------|
| Decode | Read register 'f' | Process Data | No operation |

If skip:

| Q1           | Q2           | Q3           | Q4           |
|--------------|--------------|--------------|--------------|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1           | Q2           | Q3           | Q4           |
|--------------|--------------|--------------|--------------|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

**Example:**

```

HERE    CPFSEQ REG, 0
NEQUAL  :
EQUAL   :
```

Before Instruction

```

PC Address = HERE
W          = ?
REG        = ?
```

After Instruction

```

If REG     = W;
PC         = Address (EQUAL)
If REG     ≠ W;
PC         = Address (NEQUAL)
```

| CPFSGT           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | Compare f with W, skip if f > W |      |  |  |      |      |      |      |
|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------|------|--|--|------|------|------|------|
| Syntax:          | CPFSGT f {,a}                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                                 |      |  |  |      |      |      |      |
| Operands:        | $0 \leq f \leq 255$<br>$a \in [0,1]$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                                 |      |  |  |      |      |      |      |
| Operation:       | (f) − (W),<br>skip if (f) > (W)<br>(unsigned comparison)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                                 |      |  |  |      |      |      |      |
| Status Affected: | None                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                                 |      |  |  |      |      |      |      |
| Encoding:        | <table border="1"><tr><td>0110</td><td>010a</td><td>ffff</td><td>ffff</td></tr></table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                                 |      |  |  | 0110 | 010a | ffff | ffff |
| 0110             | 010a                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | ffff                            | ffff |  |  |      |      |      |      |
| Description:     | <p>Compares the contents of data memory location 'f' to the contents of the W by performing an unsigned subtraction. If the contents of 'f' are greater than the contents of WREG, then the fetched instruction is discarded and a NOP is executed instead, making this a 2-cycle instruction.</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <a href="#">Section 42.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</a> for details.</p> |                                 |      |  |  |      |      |      |      |
| Words:           | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                                 |      |  |  |      |      |      |      |
| Cycles:          | 1(2)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                                 |      |  |  |      |      |      |      |
|                  | <b>Note:</b> 3 cycles if skip and followed by a 2-word instruction.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                                 |      |  |  |      |      |      |      |

Q Cycle Activity:

| Q1     | Q2                | Q3           | Q4           |
|--------|-------------------|--------------|--------------|
| Decode | Read register 'f' | Process Data | No operation |

If skip:

| Q1           | Q2           | Q3           | Q4           |
|--------------|--------------|--------------|--------------|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1           | Q2           | Q3           | Q4           |
|--------------|--------------|--------------|--------------|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

**Example:**

```

HERE    CPFSGT REG, 0
NGREATER :
GREATER  :
```

Before Instruction

```

PC       = Address (HERE)
W        = ?
```

After Instruction

```

If REG   > W;
PC       = Address (GREATER)
If REG   ≤ W;
PC       = Address (NGREATER)
```

## CPFSLT Compare f with W, skip if f < W

|                  |                                                                                                                                                                                                                                                                                                                                                                                            |      |      |      |      |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|------|------|------|
| Syntax:          | CPFSLT f {,a}                                                                                                                                                                                                                                                                                                                                                                              |      |      |      |      |
| Operands:        | $0 \leq f \leq 255$<br>$a \in [0,1]$                                                                                                                                                                                                                                                                                                                                                       |      |      |      |      |
| Operation:       | (f) – (W),<br>skip if (f) < (W)<br>(unsigned comparison)                                                                                                                                                                                                                                                                                                                                   |      |      |      |      |
| Status Affected: | None                                                                                                                                                                                                                                                                                                                                                                                       |      |      |      |      |
| Encoding:        | <table border="1"><tr><td>0110</td><td>000a</td><td>ffff</td><td>ffff</td></tr></table>                                                                                                                                                                                                                                                                                                    | 0110 | 000a | ffff | ffff |
| 0110             | 000a                                                                                                                                                                                                                                                                                                                                                                                       | ffff | ffff |      |      |
| Description:     | Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction. If the contents of 'f' are less than the contents of W, then the fetched instruction is discarded and a NOP is executed instead, making this a 2-cycle instruction.<br>If 'a' is '0', the Access Bank is selected.<br>If 'a' is '1', the BSR is used to select the GPR bank. |      |      |      |      |
| Words:           | 1                                                                                                                                                                                                                                                                                                                                                                                          |      |      |      |      |
| Cycles:          | 1(2)<br><b>Note:</b> 3 cycles if skip and followed by a 2-word instruction.                                                                                                                                                                                                                                                                                                                |      |      |      |      |

### Q Cycle Activity:

| Q1     | Q2                | Q3           | Q4           |
|--------|-------------------|--------------|--------------|
| Decode | Read register 'f' | Process Data | No operation |

### If skip:

| Q1           | Q2           | Q3           | Q4           |
|--------------|--------------|--------------|--------------|
| No operation | No operation | No operation | No operation |

### If skip and followed by 2-word instruction:

| Q1           | Q2           | Q3           | Q4           |
|--------------|--------------|--------------|--------------|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

**Example:**

```

HERE    CPFSLT REG, 1
NLESS   :
LESS    :
```

### Before Instruction

PC = Address (HERE)  
W = ?

### After Instruction

If REG < W;  
PC = Address (LESS)  
If REG ≥ W;  
PC = Address (NLESS)

## DAW Decimal Adjust W Register

|                  |                                                                                                                                                                                                        |      |      |      |      |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|------|------|------|
| Syntax:          | DAW                                                                                                                                                                                                    |      |      |      |      |
| Operands:        | None                                                                                                                                                                                                   |      |      |      |      |
| Operation:       | If [W<3:0> > 9] or [DC = 1] then<br>(W<3:0>) + 6 → W<3:0>;<br>else<br>(W<3:0>) → W<3:0>;<br><br>If [W<7:4> + DC > 9] or [C = 1] then<br>(W<7:4>) + 6 + DC → W<7:4>;<br>else<br>(W<7:4>) + DC → W<7:4>; |      |      |      |      |
| Status Affected: | C                                                                                                                                                                                                      |      |      |      |      |
| Encoding:        | <table border="1"><tr><td>0000</td><td>0000</td><td>0000</td><td>0111</td></tr></table>                                                                                                                | 0000 | 0000 | 0000 | 0111 |
| 0000             | 0000                                                                                                                                                                                                   | 0000 | 0111 |      |      |
| Description:     | DAW adjusts the 8-bit value in W, resulting from the earlier addition of two variables (each in packed BCD format) and produces a correct packed BCD result.                                           |      |      |      |      |
| Words:           | 1                                                                                                                                                                                                      |      |      |      |      |
| Cycles:          | 1                                                                                                                                                                                                      |      |      |      |      |

### Q Cycle Activity:

| Q1     | Q2              | Q3           | Q4      |
|--------|-----------------|--------------|---------|
| Decode | Read register W | Process Data | Write W |

### Example1:

#### DAW

### Before Instruction

W = A5h  
C = 0  
DC = 0

### After Instruction

W = 05h  
C = 1  
DC = 0

### Example 2:

### Before Instruction

W = CEh  
C = 0  
DC = 0

### After Instruction

W = 34h  
C = 1  
DC = 0

## DECF Decrement f

Syntax: `DECF f{,d{,a}}`

Operands:  $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

Operation:  $(f) - 1 \rightarrow \text{dest}$

Status Affected: C, DC, N, OV, Z

Encoding: 

|      |      |      |      |
|------|------|------|------|
| 0000 | 01da | ffff | ffff |
|------|------|------|------|

Description: Decrement register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default).  
 If 'a' is '0', the Access Bank is selected.  
 If 'a' is '1', the BSR is used to select the GPR bank.  
 If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh). See [Section 42.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"](#) for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1     | Q2                | Q3           | Q4                   |
|--------|-------------------|--------------|----------------------|
| Decode | Read register 'f' | Process Data | Write to destination |

**Example:** `DECF CNT, 1, 0`

Before Instruction

CNT = 01h  
 Z = 0

After Instruction

CNT = 00h  
 Z = 1

## DECFSZ Decrement f, skip if 0

Syntax: `DECFSZ f{,d{,a}}`

Operands:  $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

Operation:  $(f) - 1 \rightarrow \text{dest}$ ,  
 skip if result = 0

Status Affected: None

Encoding: 

|      |      |      |      |
|------|------|------|------|
| 0010 | 11da | ffff | ffff |
|------|------|------|------|

Description: The contents of register 'f' are decremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).  
 If the result is '0', the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it a 2-cycle instruction.  
 If 'a' is '0', the Access Bank is selected.  
 If 'a' is '1', the BSR is used to select the GPR bank.  
 If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh). See [Section 42.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"](#) for details.

Words: 1

Cycles: 1(2)

**Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1     | Q2                | Q3           | Q4                   |
|--------|-------------------|--------------|----------------------|
| Decode | Read register 'f' | Process Data | Write to destination |

If skip:

| Q1           | Q2           | Q3           | Q4           |
|--------------|--------------|--------------|--------------|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1           | Q2           | Q3           | Q4           |
|--------------|--------------|--------------|--------------|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

**Example:** `HERE DECFSZ CNT, 1, 1  
 GOTO LOOP  
 CONTINUE`

Before Instruction

PC = Address (HERE)

After Instruction

CNT = CNT - 1  
 If CNT = 0;  
 PC = Address (CONTINUE)  
 If CNT  $\neq$  0;  
 PC = Address (HERE + 2)



| DCFSNZ           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Decrement f, skip if not 0 |      |  |  |      |      |      |      |
|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------|------|--|--|------|------|------|------|
| Syntax:          | DCFSNZ f {,d {,a}}                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                            |      |  |  |      |      |      |      |
| Operands:        | $0 \leq f \leq 255$<br>$d \in [0,1]$<br>$a \in [0,1]$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                            |      |  |  |      |      |      |      |
| Operation:       | (f) – 1 → dest,<br>skip if result ≠ 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                            |      |  |  |      |      |      |      |
| Status Affected: | None                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                            |      |  |  |      |      |      |      |
| Encoding:        | <table><tr><td>0100</td><td>11da</td><td>ffff</td><td>ffff</td></tr></table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                            |      |  |  | 0100 | 11da | ffff | ffff |
| 0100             | 11da                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | ffff                       | ffff |  |  |      |      |      |      |
| Description:     | <p>The contents of register 'f' are decremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). If the result is not '0', the next instruction, which is already fetched, is discarded and a <code>NOP</code> is executed instead, making it a 2-cycle instruction.</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <a href="#">Section 42.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</a> for details.</p> |                            |      |  |  |      |      |      |      |
| Words:           | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                            |      |  |  |      |      |      |      |
| Cycles:          | 1(2)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                            |      |  |  |      |      |      |      |
| <b>Note:</b>     | 3 cycles if skip and followed by a 2-word instruction.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                            |      |  |  |      |      |      |      |

Q Cycle Activity:

| Q1     | Q2                | Q3           | Q4                   |
|--------|-------------------|--------------|----------------------|
| Decode | Read register 'f' | Process Data | Write to destination |

If skip:

| Q1           | Q2           | Q3           | Q4           |
|--------------|--------------|--------------|--------------|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1           | Q2           | Q3           | Q4           |
|--------------|--------------|--------------|--------------|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

**Example:**

```

HERE    DCFSNZ  TEMP, 1, 0
ZERO    :
NZERO   :
```

Before Instruction

TEMP = ?

After Instruction

```

TEMP    = TEMP – 1,
If TEMP = 0;
PC      = Address (ZERO)
If TEMP ≠ 0;
PC      = Address (NZERO)
```

| GOTO             |                                                                                                                                                                                                                                                 | Unconditional Branch |             |  |  |      |      |             |             |      |                |           |             |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------|-------------|--|--|------|------|-------------|-------------|------|----------------|-----------|-------------|
| Syntax:          | GOTO k                                                                                                                                                                                                                                          |                      |             |  |  |      |      |             |             |      |                |           |             |
| Operands:        | $0 \leq k \leq 1048575$                                                                                                                                                                                                                         |                      |             |  |  |      |      |             |             |      |                |           |             |
| Operation:       | $k \rightarrow PC<20:1>$                                                                                                                                                                                                                        |                      |             |  |  |      |      |             |             |      |                |           |             |
| Status Affected: | None                                                                                                                                                                                                                                            |                      |             |  |  |      |      |             |             |      |                |           |             |
| Encoding:        | <table border="1"><tr><td>1110</td><td>1111</td><td><math>k_7 k k k</math></td><td><math>k k k k_0</math></td></tr><tr><td>1111</td><td><math>k_{19} k k k</math></td><td><math>k k k k</math></td><td><math>k k k k_8</math></td></tr></table> |                      |             |  |  | 1110 | 1111 | $k_7 k k k$ | $k k k k_0$ | 1111 | $k_{19} k k k$ | $k k k k$ | $k k k k_8$ |
| 1110             | 1111                                                                                                                                                                                                                                            | $k_7 k k k$          | $k k k k_0$ |  |  |      |      |             |             |      |                |           |             |
| 1111             | $k_{19} k k k$                                                                                                                                                                                                                                  | $k k k k$            | $k k k k_8$ |  |  |      |      |             |             |      |                |           |             |
| Description:     | <p>GOTO allows an unconditional branch anywhere within entire 2-Mbyte memory range. The 20-bit value 'k' is loaded into PC&lt;20:1&gt;.</p> <p>GOTO is always a 2-cycle instruction.</p>                                                        |                      |             |  |  |      |      |             |             |      |                |           |             |
| Words:           | 2                                                                                                                                                                                                                                               |                      |             |  |  |      |      |             |             |      |                |           |             |
| Cycles:          | 2                                                                                                                                                                                                                                               |                      |             |  |  |      |      |             |             |      |                |           |             |

Q Cycle Activity:

| Q1           | Q2                     | Q3           | Q4                                  |
|--------------|------------------------|--------------|-------------------------------------|
| Decode       | Read literal 'k'<7:0>. | No operation | Read literal 'k'<19:8>, Write to PC |
| No operation | No operation           | No operation | No operation                        |

**Example:** GOTO THERE

After Instruction

PC = Address (THERE)

## INCF Increment f

|                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |      |      |      |      |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|------|------|------|
| Syntax:           | INCF f{,d{,a}}                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |      |      |      |      |
| Operands:         | $0 \leq f \leq 255$<br>$d \in [0,1]$<br>$a \in [0,1]$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |      |      |      |      |
| Operation:        | $(f) + 1 \rightarrow \text{dest}$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |      |      |      |      |
| Status Affected:  | C, DC, N, OV, Z                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |      |      |      |      |
| Encoding:         | <table border="1"><tr><td>0010</td><td>10da</td><td>ffff</td><td>ffff</td></tr></table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | 0010 | 10da | ffff | ffff |
| 0010              | 10da                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | ffff | ffff |      |      |
| Description:      | <p>The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <a href="#">Section 42.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</a> for details.</p> |      |      |      |      |
| Words:            | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |      |      |      |      |
| Cycles:           | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |      |      |      |      |
| Q Cycle Activity: |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |      |      |      |      |

| Q1     | Q2                | Q3           | Q4                   |
|--------|-------------------|--------------|----------------------|
| Decode | Read register 'f' | Process Data | Write to destination |

**Example:** INCF CNT, 1, 0

|                    |   |     |
|--------------------|---|-----|
| Before Instruction |   |     |
| CNT                | = | FFh |
| Z                  | = | 0   |
| C                  | = | ?   |
| DC                 | = | ?   |
| After Instruction  |   |     |
| CNT                | = | 00h |
| Z                  | = | 1   |
| C                  | = | 1   |
| DC                 | = | 1   |

## INCFSZ Increment f, skip if 0

|                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |      |      |  |      |      |      |      |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|------|--|------|------|------|------|
| Syntax:          | INCFSZ f {,d {,a}}                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |      |      |  |      |      |      |      |
| Operands:        | $0 \leq f \leq 255$<br>$d \in [0,1]$<br>$a \in [0,1]$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |      |      |  |      |      |      |      |
| Operation:       | $(f) + 1 \rightarrow \text{dest}$ ,<br>skip if result = 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |      |      |  |      |      |      |      |
| Status Affected: | None                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |      |      |  |      |      |      |      |
| Encoding:        | <table border="1"><tr><td>0011</td><td>11da</td><td>ffff</td><td>ffff</td></tr></table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |      |      |  | 0011 | 11da | ffff | ffff |
| 0011             | 11da                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | ffff | ffff |  |      |      |      |      |
| Description:     | <p>The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). If the result is '0', the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it a 2-cycle instruction.</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <a href="#">Section 42.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</a> for details.</p> |      |      |  |      |      |      |      |
| Words:           | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |      |      |  |      |      |      |      |
| Cycles:          | 1(2)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |      |      |  |      |      |      |      |
|                  | <b>Note:</b> 3 cycles if skip and followed by a 2-word instruction.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |      |      |  |      |      |      |      |

Q Cycle Activity:

| Q1     | Q2                | Q3           | Q4                   |
|--------|-------------------|--------------|----------------------|
| Decode | Read register 'f' | Process Data | Write to destination |

If skip:

| Q1           | Q2           | Q3           | Q4           |
|--------------|--------------|--------------|--------------|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1           | Q2           | Q3           | Q4           |
|--------------|--------------|--------------|--------------|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

**Example:** HERE INCFSZ CNT, 1, 0  
NZERO :  
ZERO :

|                    |   |                 |
|--------------------|---|-----------------|
| Before Instruction |   |                 |
| PC                 | = | Address (HERE)  |
| After Instruction  |   |                 |
| CNT                | = | CNT + 1         |
| If CNT             | = | 0;              |
| PC                 | = | Address (ZERO)  |
| If CNT             | ≠ | 0;              |
| PC                 | = | Address (NZERO) |

| INFSNZ           |                                                                                                                                                                                                                                                                                                                                                                                       |  |  |  | Increment f, skip if not 0 |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|--|--|----------------------------|
| Syntax:          | INFSNZ                                                                                                                                                                                                                                                                                                                                                                                |  |  |  | f {,d {,a}}                |
| Operands:        | 0 ≤ f ≤ 255                                                                                                                                                                                                                                                                                                                                                                           |  |  |  |                            |
|                  | d ∈ [0,1]                                                                                                                                                                                                                                                                                                                                                                             |  |  |  |                            |
|                  | a ∈ [0,1]                                                                                                                                                                                                                                                                                                                                                                             |  |  |  |                            |
| Operation:       | (f) + 1 → dest,                                                                                                                                                                                                                                                                                                                                                                       |  |  |  | skip if result ≠ 0         |
| Status Affected: | None                                                                                                                                                                                                                                                                                                                                                                                  |  |  |  |                            |
| Encoding:        | 0100                                                                                                                                                                                                                                                                                                                                                                                  |  |  |  | 10da    ffff    ffff       |
| Description:     | The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). If the result is not '0', the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it a 2-cycle instruction.                                                                    |  |  |  |                            |
|                  | If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See <a href="#">Section 42.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</a> for details. |  |  |  |                            |
| Words:           | 1                                                                                                                                                                                                                                                                                                                                                                                     |  |  |  |                            |
| Cycles:          | 1(2)                                                                                                                                                                                                                                                                                                                                                                                  |  |  |  |                            |
|                  | <b>Note:</b> 3 cycles if skip and followed by a 2-word instruction.                                                                                                                                                                                                                                                                                                                   |  |  |  |                            |

Q Cycle Activity:

| Q1     | Q2                | Q3           | Q4                   |
|--------|-------------------|--------------|----------------------|
| Decode | Read register 'f' | Process Data | Write to destination |

If skip:

| Q1           | Q2           | Q3           | Q4           |
|--------------|--------------|--------------|--------------|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1           | Q2           | Q3           | Q4           |
|--------------|--------------|--------------|--------------|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

**Example:**

```

HERE    INFSNZ REG, 1, 0
ZERO
NZERO

```

Before Instruction

PC = Address (HERE)

After Instruction

REG = REG + 1

If REG  $\neq$  0;

PC = Address (NZERO)

If REG = 0;

PC = Address (ZERO)

| IORLW            |                                                                                         | Inclusive OR literal with W |      |  |      |      |      |      |
|------------------|-----------------------------------------------------------------------------------------|-----------------------------|------|--|------|------|------|------|
| Syntax:          | IORLW k                                                                                 |                             |      |  |      |      |      |      |
| Operands:        | $0 \leq k \leq 255$                                                                     |                             |      |  |      |      |      |      |
| Operation:       | $(W) .OR. k \rightarrow W$                                                              |                             |      |  |      |      |      |      |
| Status Affected: | N, Z                                                                                    |                             |      |  |      |      |      |      |
| Encoding:        | <table border="1"><tr><td>0000</td><td>1001</td><td>kkkk</td><td>kkkk</td></tr></table> |                             |      |  | 0000 | 1001 | kkkk | kkkk |
| 0000             | 1001                                                                                    | kkkk                        | kkkk |  |      |      |      |      |
| Description:     | The contents of W are ORed with the 8-bit literal 'k'. The result is placed in W.       |                             |      |  |      |      |      |      |
| Words:           | 1                                                                                       |                             |      |  |      |      |      |      |
| Cycles:          | 1                                                                                       |                             |      |  |      |      |      |      |

Q Cycle Activity:

| Q1     | Q2               | Q3           | Q4         |
|--------|------------------|--------------|------------|
| Decode | Read literal 'k' | Process Data | Write to W |

**Example:** IORLW 35h

Before Instruction

W = 9Ah

After Instruction

W = BFh

| IORWF             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Inclusive OR W with f |              |                      |      |      |      |      |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|--------------|----------------------|------|------|------|------|
| Syntax:           | IORWF    f {,d {,a}}                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                       |              |                      |      |      |      |      |
| Operands:         | $0 \leq f \leq 255$<br>$d \in [0,1]$<br>$a \in [0,1]$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                       |              |                      |      |      |      |      |
| Operation:        | (W) .OR. (f) → dest                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                       |              |                      |      |      |      |      |
| Status Affected:  | N, Z                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                       |              |                      |      |      |      |      |
| Encoding:         | <table border="1"><tr><td>0001</td><td>00da</td><td>ffff</td><td>ffff</td></tr></table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                       |              |                      | 0001 | 00da | ffff | ffff |
| 0001              | 00da                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | ffff                  | ffff         |                      |      |      |      |      |
| Description:      | <p>Inclusive OR W with register 'f'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).</p> <p>If 'a' is '0', the Access Bank is selected.</p> <p>If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <a href="#">Section 42.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</a> for details.</p> |                       |              |                      |      |      |      |      |
| Words:            | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                       |              |                      |      |      |      |      |
| Cycles:           | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                       |              |                      |      |      |      |      |
| Q Cycle Activity: |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                       |              |                      |      |      |      |      |
|                   | Q1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | Q2                    | Q3           | Q4                   |      |      |      |      |
|                   | Decode                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Read register 'f'     | Process Data | Write to destination |      |      |      |      |

**Example:** IORWF RESULT, 0, 1

Before Instruction

RESULT = 13h

W = 91h

After Instruction

RESULT = 13h

W = 93h

| LFSR              |                                                                                                                                                                    | Load FSR             |      |              |      |                                |                     |      |      |      |                    |      |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------|------|--------------|------|--------------------------------|---------------------|------|------|------|--------------------|------|
| Syntax:           | LFSR f, k                                                                                                                                                          |                      |      |              |      |                                |                     |      |      |      |                    |      |
| Operands:         | $0 \leq f \leq 2$<br>$0 \leq k \leq 16383$                                                                                                                         |                      |      |              |      |                                |                     |      |      |      |                    |      |
| Operation:        | $k \rightarrow \text{FSRf}$                                                                                                                                        |                      |      |              |      |                                |                     |      |      |      |                    |      |
| Status Affected:  | None                                                                                                                                                               |                      |      |              |      |                                |                     |      |      |      |                    |      |
| Encoding:         | <table><tr><td>1110</td><td>1110</td><td>00k<sub>13</sub>k</td><td>kkkk</td></tr><tr><td>1111</td><td>0000</td><td>k<sub>7</sub>kkk</td><td>kkkk</td></tr></table> |                      |      |              | 1110 | 1110                           | 00k <sub>13</sub> k | kkkk | 1111 | 0000 | k <sub>7</sub> kkk | kkkk |
| 1110              | 1110                                                                                                                                                               | 00k <sub>13</sub> k  | kkkk |              |      |                                |                     |      |      |      |                    |      |
| 1111              | 0000                                                                                                                                                               | k <sub>7</sub> kkk   | kkkk |              |      |                                |                     |      |      |      |                    |      |
| Description:      | The 14-bit literal 'k' is loaded into the File Select Register pointed to by 'f'.                                                                                  |                      |      |              |      |                                |                     |      |      |      |                    |      |
| Words:            | 2                                                                                                                                                                  |                      |      |              |      |                                |                     |      |      |      |                    |      |
| Cycles:           | 2                                                                                                                                                                  |                      |      |              |      |                                |                     |      |      |      |                    |      |
| Q Cycle Activity: |                                                                                                                                                                    |                      |      |              |      |                                |                     |      |      |      |                    |      |
| Q1                |                                                                                                                                                                    | Q2                   |      | Q3           |      | Q4                             |                     |      |      |      |                    |      |
| Decode            |                                                                                                                                                                    | Read literal 'k' MSB |      | Process Data |      | Write literal 'k' MSB to FSRfH |                     |      |      |      |                    |      |
| Decode            |                                                                                                                                                                    | Read literal 'k' LSB |      | Process Data |      | Write literal 'k' to FSRfL     |                     |      |      |      |                    |      |

**Example:** LFSR 2, 3ABh

After Instruction

FSR2H = 03h

FSR2L = ABh

## MOVF

## Move f

Syntax: MOVF f {,d {,a}}

Operands:  $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

Operation:  $f \rightarrow \text{dest}$

Status Affected: N, Z

Encoding: 

|      |      |      |      |
|------|------|------|------|
| 0101 | 00da | ffff | ffff |
|------|------|------|------|

Description: The contents of register 'f' are moved to a destination dependent upon the status of 'd'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). Location 'f' can be anywhere in the 256-byte bank.  
 If 'a' is '0', the Access Bank is selected.  
 If 'a' is '1', the BSR is used to select the GPR bank.  
 If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh). See [Section 42.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"](#) for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1     | Q2                | Q3           | Q4      |
|--------|-------------------|--------------|---------|
| Decode | Read register 'f' | Process Data | Write W |

**Example:** MOVF REG, 0, 0

Before Instruction

REG = 22h  
 W = FFh

After Instruction

REG = 22h  
 W = 22h

## MOVFF

## Move f to f

Syntax: MOVFF  $f_s, f_d$

Operands:  $0 \leq f_s \leq 4095$   
 $0 \leq f_d \leq 4095$

Operation:  $(f_s) \rightarrow f_d$

Status Affected: None

Encoding: 

|      |      |      |                   |
|------|------|------|-------------------|
| 1100 | ffff | ffff | ffff <sub>s</sub> |
| 1111 | ffff | ffff | ffff <sub>d</sub> |

Description: The contents of source register 'f<sub>s</sub>' are moved to destination register 'f<sub>d</sub>'. Location of source 'f<sub>s</sub>' can be anywhere in the 4096-byte data space (000h to FFFh) and location of destination 'f<sub>d</sub>' can also be anywhere from 000h to FFFh.

MOVFF has curtailed the source and destination range to the lower 4 Kbyte space of memory (Banks 1 through 15). For everything else, use MOVFFL.

Words: 2

Cycles: 2 (3)

Q Cycle Activity:

| Q1     | Q2                            | Q3           | Q4                        |
|--------|-------------------------------|--------------|---------------------------|
| Decode | Read register 'f' (src)       | Process Data | No operation              |
| Decode | No operation<br>No dummy read | No operation | Write register 'f' (dest) |

**Example:** MOVFF REG1, REG2

Before Instruction

REG1 = 33h  
 REG2 = 11h

After Instruction

REG1 = 33h  
 REG2 = 33h

## MOVFFL Move f to f (Long Range)

|                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax:           | MOVFFL $f_s, f_d$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Operands:         | $0 \leq f_s \leq 16383$<br>$0 \leq f_d \leq 16383$                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Operation:        | $(f_s) \rightarrow f_d$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| Status Affected:  | None                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Encoding:         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 1st word          | 0000                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 2nd word          | 1111                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 3rd word          | 1111                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Description:      | <p>The contents of source register '<math>f_s</math>' are moved to destination register '<math>f_d</math>'. Location of source '<math>f_s</math>' can be anywhere in the 16 Kbyte data space (0000h to 3FFFh). Either source or destination can be W (a useful special situation). MOVFFL is particularly useful for transferring a data memory location to a peripheral register (such as the transmit buffer or an I/O port). The MOVFFL instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.</p> |
| Words:            | 3                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Cycles:           | 3                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Q Cycle Activity: |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

| Q1     | Q2                            | Q3           | Q4                              |
|--------|-------------------------------|--------------|---------------------------------|
| Decode | No operation                  | No operation | No operation                    |
| Decode | Read register ' $f_s$ ' (src) | Process data | No operation                    |
| Decode | No operation<br>No dummy read | No operation | Write register ' $f_d$ ' (dest) |

**Example:** MOVFFL 2000h, 200Ah

Before Instruction

Contents of 2000h = 33h  
Contents of 200Ah = 11h

After Instruction

Contents of 2000h = 33h  
Contents of 200Ah = 33h

## MOVLB Move literal to BSR

| Syntax:           | MOVLW k                                                                                                                                                                     |              |                          |      |      |        |                  |              |                          |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|--------------------------|------|------|--------|------------------|--------------|--------------------------|
| Operands:         | $0 \leq k \leq 63$                                                                                                                                                          |              |                          |      |      |        |                  |              |                          |
| Operation:        | $k \rightarrow \text{BSR}$                                                                                                                                                  |              |                          |      |      |        |                  |              |                          |
| Status Affected:  | None                                                                                                                                                                        |              |                          |      |      |        |                  |              |                          |
| Encoding:         | <table border="1"><tr><td>0000</td><td>0001</td><td>00kk</td><td>kkkk</td></tr></table>                                                                                     | 0000         | 0001                     | 00kk | kkkk |        |                  |              |                          |
| 0000              | 0001                                                                                                                                                                        | 00kk         | kkkk                     |      |      |        |                  |              |                          |
| Description:      | The 6-bit literal 'k' is loaded into the Bank Select Register (BSR<5:0>). The value of BSR<7:6> always remains '0'.                                                         |              |                          |      |      |        |                  |              |                          |
| Words:            | 1                                                                                                                                                                           |              |                          |      |      |        |                  |              |                          |
| Cycles:           | 1                                                                                                                                                                           |              |                          |      |      |        |                  |              |                          |
| Q Cycle Activity: |                                                                                                                                                                             |              |                          |      |      |        |                  |              |                          |
|                   | <table><tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr><tr><td>Decode</td><td>Read literal 'k'</td><td>Process Data</td><td>Write literal 'k' to BSR</td></tr></table> | Q1           | Q2                       | Q3   | Q4   | Decode | Read literal 'k' | Process Data | Write literal 'k' to BSR |
| Q1                | Q2                                                                                                                                                                          | Q3           | Q4                       |      |      |        |                  |              |                          |
| Decode            | Read literal 'k'                                                                                                                                                            | Process Data | Write literal 'k' to BSR |      |      |        |                  |              |                          |

**Example:** MOVLB 5

Before Instruction

BSR Register = 02h

After Instruction

BSR Register = 05h

## MOVLW Move literal to W

Syntax: MOVLW k

Operands:  $0 \leq k \leq 255$

Operation:  $k \rightarrow W$

Status Affected: None

Encoding:

|      |      |      |      |
|------|------|------|------|
| 0000 | 1110 | kkkk | kkkk |
|------|------|------|------|

Description: The 8-bit literal 'k' is loaded into W.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1     | Q2               | Q3           | Q4         |
|--------|------------------|--------------|------------|
| Decode | Read literal 'k' | Process Data | Write to W |

**Example:** MOVLW 5Ah

After Instruction

W = 5Ah

## MOVWF Move W to f

Syntax: MOVWF f{,a}

Operands:  $0 \leq f \leq 255$   
 $a \in [0,1]$

Operation:  $(W) \rightarrow f$

Status Affected: None

Encoding:

|      |      |      |      |
|------|------|------|------|
| 0110 | 111a | ffff | ffff |
|------|------|------|------|

Description: Move data from W to register 'f'. Location 'f' can be anywhere in the 256-byte bank.

If 'a' is '0', the Access Bank is selected.  
If 'a' is '1', the BSR is used to select the GPR bank.  
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh). See [Section 42.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"](#) for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1     | Q2                | Q3           | Q4                 |
|--------|-------------------|--------------|--------------------|
| Decode | Read register 'f' | Process Data | Write register 'f' |

**Example:** MOVWF REG, 0

Before Instruction

W = 4Fh

REG = FFh

After Instruction

W = 4Fh

REG = 4Fh

| MULLW            | Multiply literal with W                                                                                                                                                                                                                                                                                                                                                         |      |      |      |      |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|------|------|------|
| Syntax:          | MULLW   k                                                                                                                                                                                                                                                                                                                                                                       |      |      |      |      |
| Operands:        | 0 ≤ k ≤ 255                                                                                                                                                                                                                                                                                                                                                                     |      |      |      |      |
| Operation:       | (W) × k → PRODH:PRODL                                                                                                                                                                                                                                                                                                                                                           |      |      |      |      |
| Status Affected: | None                                                                                                                                                                                                                                                                                                                                                                            |      |      |      |      |
| Encoding:        | <table><tr><td>0000</td><td>1101</td><td>kkkk</td><td>kkkk</td></tr></table>                                                                                                                                                                                                                                                                                                    | 0000 | 1101 | kkkk | kkkk |
| 0000             | 1101                                                                                                                                                                                                                                                                                                                                                                            | kkkk | kkkk |      |      |
| Description:     | <p>An unsigned multiplication is carried out between the contents of W and the 8-bit literal 'k'. The 16-bit result is placed in the PRODH:PRODL register pair. PRODH contains the high byte. W is unchanged.</p> <p>None of the Status flags are affected. Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected.</p> |      |      |      |      |
| Words:           | 1                                                                                                                                                                                                                                                                                                                                                                               |      |      |      |      |
| Cycles:          | 1                                                                                                                                                                                                                                                                                                                                                                               |      |      |      |      |

Q Cycle Activity:

| Q1     | Q2               | Q3           | Q4                           |
|--------|------------------|--------------|------------------------------|
| Decode | Read literal 'k' | Process Data | Write registers PRODH: PRODL |

**Example:** MULLW 0C4h

Before Instruction

W = E2h  
PRODH = ?  
PRODL = ?

After Instruction

W = E2h  
PRODH = ADh  
PRODL = 08h

| MULWF            |                                                                                                                                                                                                                                                                                                                                                                                                                                                 | Multiply W with f |      |  |  |      |      |      |      |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|------|--|--|------|------|------|------|
| Syntax:          | MULWF    f {,a}                                                                                                                                                                                                                                                                                                                                                                                                                                 |                   |      |  |  |      |      |      |      |
| Operands:        | $0 \leq f \leq 255$<br>$a \in [0,1]$                                                                                                                                                                                                                                                                                                                                                                                                            |                   |      |  |  |      |      |      |      |
| Operation:       | $(W) \times (f) \rightarrow \text{PRODH:PRODL}$                                                                                                                                                                                                                                                                                                                                                                                                 |                   |      |  |  |      |      |      |      |
| Status Affected: | None                                                                                                                                                                                                                                                                                                                                                                                                                                            |                   |      |  |  |      |      |      |      |
| Encoding:        | <table border="1"><tr><td>0000</td><td>001a</td><td>ffff</td><td>ffff</td></tr></table>                                                                                                                                                                                                                                                                                                                                                         |                   |      |  |  | 0000 | 001a | ffff | ffff |
| 0000             | 001a                                                                                                                                                                                                                                                                                                                                                                                                                                            | ffff              | ffff |  |  |      |      |      |      |
| Description:     | <p>An unsigned multiplication is carried out between the contents of W and the register file location 'f'. The 16-bit result is stored in the PRODH:PRODL register pair. PRODH contains the high byte. Both W and 'f' are unchanged.</p> <p>None of the Status flags are affected. Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected.</p> <p>If 'a' is '0', the Access Bank is</p> |                   |      |  |  |      |      |      |      |

Q Cycle Activity:

| Q1     | Q2                | Q3           | Q4                           |
|--------|-------------------|--------------|------------------------------|
| Decode | Read register 'f' | Process Data | Write registers PRODH: PRODL |

**Example:** MULWF REG, 1

Before Instruction

W = C4h  
REG = B5h  
PRODH = ?  
PRODL = ?

After Instruction

W = C4h  
REG = B5h  
PRODH = 8Ah  
PRODL = 94h



## NEGF

## Negate f

|                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                   |              |                    |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|--------------|--------------------|
| Syntax:           | NEGF f {,a}                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                   |              |                    |
| Operands:         | $0 \leq f \leq 255$<br>$a \in [0,1]$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                   |              |                    |
| Operation:        | $(\bar{f}) + 1 \rightarrow f$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                   |              |                    |
| Status Affected:  | N, OV, C, DC, Z                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                   |              |                    |
| Encoding:         | 0110                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | 110a              | ffff         | ffff               |
| Description:      | <p>Location 'f' is negated using two's complement. The result is placed in the data memory location 'f'.</p> <p>If 'a' is '0', the Access Bank is selected.</p> <p>If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <a href="#">Section 42.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</a> for details.</p> |                   |              |                    |
| Words:            | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                   |              |                    |
| Cycles:           | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                   |              |                    |
| Q Cycle Activity: |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                   |              |                    |
|                   | Q1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | Q2                | Q3           | Q4                 |
|                   | Decode                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Read register 'f' | Process Data | Write register 'f' |

**Example:** NEGF REG, 1

Before Instruction

REG = 0011 1010 [3Ah]

After Instruction

REG = 1100 0110 [C6h]

## NOP

## No Operation

|                   |                                                                                                                                                             |              |              |      |      |        |              |              |              |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|--------------|------|------|--------|--------------|--------------|--------------|
| Syntax:           | NOP                                                                                                                                                         |              |              |      |      |        |              |              |              |
| Operands:         | None                                                                                                                                                        |              |              |      |      |        |              |              |              |
| Operation:        | No operation                                                                                                                                                |              |              |      |      |        |              |              |              |
| Status Affected:  | None                                                                                                                                                        |              |              |      |      |        |              |              |              |
| Encoding:         | <table><tr><td>0000</td><td>0000</td><td>0000</td><td>0000</td></tr><tr><td>1111</td><td>xxxx</td><td>xxxx</td><td>xxxx</td></tr></table>                   | 0000         | 0000         | 0000 | 0000 | 1111   | xxxx         | xxxx         | xxxx         |
| 0000              | 0000                                                                                                                                                        | 0000         | 0000         |      |      |        |              |              |              |
| 1111              | xxxx                                                                                                                                                        | xxxx         | xxxx         |      |      |        |              |              |              |
| Description:      | No operation.                                                                                                                                               |              |              |      |      |        |              |              |              |
| Words:            | 1                                                                                                                                                           |              |              |      |      |        |              |              |              |
| Cycles:           | 1                                                                                                                                                           |              |              |      |      |        |              |              |              |
| Q Cycle Activity: |                                                                                                                                                             |              |              |      |      |        |              |              |              |
|                   | <table><tr><td>Q1</td><td>Q2</td><td>Q3</td><td>Q4</td></tr><tr><td>Decode</td><td>No operation</td><td>No operation</td><td>No operation</td></tr></table> | Q1           | Q2           | Q3   | Q4   | Decode | No operation | No operation | No operation |
| Q1                | Q2                                                                                                                                                          | Q3           | Q4           |      |      |        |              |              |              |
| Decode            | No operation                                                                                                                                                | No operation | No operation |      |      |        |              |              |              |

**Example:**

None.

| POP               |                                                                                                                                                                                                                                                                                        | Pop Top of Return Stack |               |              |  |  |      |      |      |      |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------|---------------|--------------|--|--|------|------|------|------|
| Syntax:           | POP                                                                                                                                                                                                                                                                                    |                         |               |              |  |  |      |      |      |      |
| Operands:         | None                                                                                                                                                                                                                                                                                   |                         |               |              |  |  |      |      |      |      |
| Operation:        | (TOS) → bit bucket                                                                                                                                                                                                                                                                     |                         |               |              |  |  |      |      |      |      |
| Status Affected:  | None                                                                                                                                                                                                                                                                                   |                         |               |              |  |  |      |      |      |      |
| Encoding:         | <table border="1"><tr><td>0000</td><td>0000</td><td>0000</td><td>0110</td></tr></table>                                                                                                                                                                                                |                         |               |              |  |  | 0000 | 0000 | 0000 | 0110 |
| 0000              | 0000                                                                                                                                                                                                                                                                                   | 0000                    | 0110          |              |  |  |      |      |      |      |
| Description:      | <p>The TOS value is pulled off the return stack and is discarded. The TOS value then becomes the previous value that was pushed onto the return stack.</p> <p>This instruction is provided to enable the user to properly manage the return stack to incorporate a software stack.</p> |                         |               |              |  |  |      |      |      |      |
| Words:            | 1                                                                                                                                                                                                                                                                                      |                         |               |              |  |  |      |      |      |      |
| Cycles:           | 1                                                                                                                                                                                                                                                                                      |                         |               |              |  |  |      |      |      |      |
| Q Cycle Activity: |                                                                                                                                                                                                                                                                                        |                         |               |              |  |  |      |      |      |      |
|                   | Q1                                                                                                                                                                                                                                                                                     | Q2                      | Q3            | Q4           |  |  |      |      |      |      |
|                   | Decode                                                                                                                                                                                                                                                                                 | No operation            | POP TOS value | No operation |  |  |      |      |      |      |

**Example:**

|      |     |  |
|------|-----|--|
| POP  |     |  |
| GOTO | NEW |  |

Before Instruction

|                      |   |         |
|----------------------|---|---------|
| TOS                  | = | 0031A2h |
| Stack (1 level down) | = | 014332h |

After Instruction

|     |   |         |
|-----|---|---------|
| TOS | = | 014332h |
| PC  | = | NEW     |

| PUSH              |                                                                                                                                                                                                                                     | Push Top of Return Stack            |      |                 |  |                 |      |      |      |      |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------|------|-----------------|--|-----------------|------|------|------|------|
| Syntax:           | PUSH                                                                                                                                                                                                                                |                                     |      |                 |  |                 |      |      |      |      |
| Operands:         | None                                                                                                                                                                                                                                |                                     |      |                 |  |                 |      |      |      |      |
| Operation:        | (PC + 2) → TOS                                                                                                                                                                                                                      |                                     |      |                 |  |                 |      |      |      |      |
| Status Affected:  | None                                                                                                                                                                                                                                |                                     |      |                 |  |                 |      |      |      |      |
| Encoding:         | <table border="1"><tr><td>0000</td><td>0000</td><td>0000</td><td>0101</td></tr></table>                                                                                                                                             |                                     |      |                 |  |                 | 0000 | 0000 | 0000 | 0101 |
| 0000              | 0000                                                                                                                                                                                                                                | 0000                                | 0101 |                 |  |                 |      |      |      |      |
| Description:      | <p>The PC + 2 is pushed onto the top of the return stack. The previous TOS value is pushed down on the stack. This instruction allows implementing a software stack by modifying TOS and then pushing it onto the return stack.</p> |                                     |      |                 |  |                 |      |      |      |      |
| Words:            | 1                                                                                                                                                                                                                                   |                                     |      |                 |  |                 |      |      |      |      |
| Cycles:           | 1                                                                                                                                                                                                                                   |                                     |      |                 |  |                 |      |      |      |      |
| Q Cycle Activity: |                                                                                                                                                                                                                                     |                                     |      |                 |  |                 |      |      |      |      |
| Q1                |                                                                                                                                                                                                                                     | Q2                                  |      | Q3              |  | Q4              |      |      |      |      |
| Decode            |                                                                                                                                                                                                                                     | PUSH<br>PC + 2 onto<br>return stack |      | No<br>operation |  | No<br>operation |      |      |      |      |

**Example:**

|      |  |  |
|------|--|--|
| PUSH |  |  |
|------|--|--|

Before Instruction

|     |   |       |
|-----|---|-------|
| TOS | = | 345Ah |
| PC  | = | 0124h |

After Instruction

|                      |   |       |
|----------------------|---|-------|
| PC                   | = | 0126h |
| TOS                  | = | 0126h |
| Stack (1 level down) | = | 345Ah |

## RCALL Relative Call

|                  |                                                                                                                                                                                                                                                                                                                                |      |      |      |      |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|------|------|------|
| Syntax:          | RCALL    n                                                                                                                                                                                                                                                                                                                     |      |      |      |      |
| Operands:        | -1024 ≤ n ≤ 1023                                                                                                                                                                                                                                                                                                               |      |      |      |      |
| Operation:       | (PC) + 2 → TOS,<br>(PC) + 2 + 2n → PC                                                                                                                                                                                                                                                                                          |      |      |      |      |
| Status Affected: | None                                                                                                                                                                                                                                                                                                                           |      |      |      |      |
| Encoding:        | <table><tr><td>1101</td><td>1nnn</td><td>nnnn</td><td>nnnn</td></tr></table>                                                                                                                                                                                                                                                   | 1101 | 1nnn | nnnn | nnnn |
| 1101             | 1nnn                                                                                                                                                                                                                                                                                                                           | nnnn | nnnn |      |      |
| Description:     | Subroutine call with a jump up to 1K from the current location. First, return address (PC + 2) is pushed onto the stack. Then, add the 2's complement number '2n' to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is a 2-cycle instruction. |      |      |      |      |
| Words:           | 1                                                                                                                                                                                                                                                                                                                              |      |      |      |      |
| Cycles:          | 2                                                                                                                                                                                                                                                                                                                              |      |      |      |      |

Q Cycle Activity:

| Q1           | Q2                                   | Q3           | Q4           |
|--------------|--------------------------------------|--------------|--------------|
| Decode       | Read literal 'n'<br>PUSH PC to stack | Process Data | Write to PC  |
| No operation | No operation                         | No operation | No operation |

**Example:**            HERE        RCALL Jump

Before Instruction

PC = Address (HERE)

After Instruction

PC = Address (Jump)

TOS = Address (HERE + 2)

## RESET Reset

|                  |                                                                              |      |      |      |      |
|------------------|------------------------------------------------------------------------------|------|------|------|------|
| Syntax:          | RESET                                                                        |      |      |      |      |
| Operands:        | None                                                                         |      |      |      |      |
| Operation:       | Reset all registers and flags that are affected by a MCLR Reset.             |      |      |      |      |
| Status Affected: | All                                                                          |      |      |      |      |
| Encoding:        | <table><tr><td>0000</td><td>0000</td><td>1111</td><td>1111</td></tr></table> | 0000 | 0000 | 1111 | 1111 |
| 0000             | 0000                                                                         | 1111 | 1111 |      |      |
| Description:     | This instruction provides a way to execute a MCLR Reset by software.         |      |      |      |      |
| Words:           | 1                                                                            |      |      |      |      |
| Cycles:          | 1                                                                            |      |      |      |      |

Q Cycle Activity:

| Q1     | Q2          | Q3           | Q4           |
|--------|-------------|--------------|--------------|
| Decode | Start Reset | No operation | No operation |

**Example:**            RESET

After Instruction

Registers = Reset Value

Flags\* = Reset Value

## RETFIE Return from Interrupt

**Syntax:** RETFIE {s}

**Operands:**  $s \in [0,1]$

**Operation:** (TOS) → PC,  
if  $s = 1$ , context is restored into WREG, STATUS, BSR, FSR0H, FSR0L, FSR1H, FSR1L, FSR2H, FSR2L, PRODH, PRODL, PCLATH and PCLATU registers from the corresponding shadow registers.

if  $s = 0$ , there is no change in status of any register.

**Status Affected:** STAT<1:0> in INTCON1 register

**Encoding:**

|      |      |      |      |
|------|------|------|------|
| 0000 | 0000 | 0001 | 000s |
|------|------|------|------|

**Description:** Return from interrupt. Stack is popped and Top-of-Stack (TOS) is loaded into the PC. Interrupts are enabled by setting either the high or low priority global interrupt enable bit. If 's' = 1, the contents of the shadow registers, WREG, STATUS, BSR, FSR0H, FSR0L, FSR1H, FSR1L, FSR2H, FSR2L, PRODH, PRODL, PCLATH and PCLATU, are loaded into corresponding registers. There are two sets of shadow registers, main context and low context. The set retrieved on RETFIE instruction execution depends on what the state of operation of the CPU was when RETFIE was executed. If 's' = 0, no update of these registers occurs (default).

**Words:** 1

**Cycles:** 2

**Q Cycle Activity:**

| Q1           | Q2           | Q3           | Q4                                    |
|--------------|--------------|--------------|---------------------------------------|
| Decode       | No operation | No operation | POP PC from stack<br>Set GIEH or GIEL |
| No operation | No operation | No operation | No operation                          |

**Example:** RETFIE 1

After Interrupt

|          |   |               |
|----------|---|---------------|
| PC       | = | TOS           |
| WREG     | = | WREG_SHAD     |
| BSR      | = | BSR_SHAD      |
| STATUS   | = | STATUS_SHAD   |
| FSR0L/H  | = | FSR0L/H_SHAD  |
| FSR1L/H  | = | FSR1L/H_SHAD  |
| FSR2L/H  | = | FSR2L/H_SHAD  |
| PRODH    | = | PROD/H_SHAD   |
| PCLATH/U | = | PCLATH/U_SHAD |

## RETLW Return literal to W

**Syntax:** RETLW k

**Operands:**  $0 \leq k \leq 255$

**Operation:**  $k \rightarrow W$ ,  
(TOS) → PC,  
PCLATU, PCLATH are unchanged

**Status Affected:** None

**Encoding:**

|      |      |      |      |
|------|------|------|------|
| 0000 | 1100 | kkkk | kkkk |
|------|------|------|------|

**Description:** W is loaded with the 8-bit literal 'k'. The Program Counter is loaded from the top of the stack (the return address). The high address latch (PCLATH) remains unchanged.

**Words:** 1

**Cycles:** 2

**Q Cycle Activity:**

| Q1           | Q2               | Q3           | Q4                               |
|--------------|------------------|--------------|----------------------------------|
| Decode       | Read literal 'k' | Process Data | POP PC from stack,<br>Write to W |
| No operation | No operation     | No operation | No operation                     |

**Example:**

```
CALL TABLE ; W contains table
               ; offset value
               ; W now has
               ; table value
:
TABLE
  ADDWF PCL ; W = offset
  RETLW k0 ; Begin table
  RETLW k1 ;
:
:
  RETLW kn ; End of table
```

**Before Instruction**

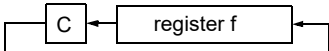
W = 07h

**After Instruction**

W = value of kn

| RETURN            |                                                                                                                                                                                                                                                                                                                    | Return from Subroutine |      |              |      |                   |      |      |  |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|------|--------------|------|-------------------|------|------|--|
| Syntax:           | RETURN {s}                                                                                                                                                                                                                                                                                                         |                        |      |              |      |                   |      |      |  |
| Operands:         | s ∈ [0,1]                                                                                                                                                                                                                                                                                                          |                        |      |              |      |                   |      |      |  |
| Operation:        | (TOS) → PC,<br>if s = 1<br>(WS) → W,<br>(STATUS) → Status,<br>(BSRS) → BSR,<br>PCLATU, PCLATH are unchanged                                                                                                                                                                                                        |                        |      |              |      |                   |      |      |  |
| Status Affected:  | None                                                                                                                                                                                                                                                                                                               |                        |      |              |      |                   |      |      |  |
| Encoding:         | <table border="1"><tr><td>0000</td><td>0000</td><td>0001</td><td>001s</td></tr></table>                                                                                                                                                                                                                            |                        |      |              | 0000 | 0000              | 0001 | 001s |  |
| 0000              | 0000                                                                                                                                                                                                                                                                                                               | 0001                   | 001s |              |      |                   |      |      |  |
| Description:      | Return from subroutine. The stack is popped and the top of the stack (TOS) is loaded into the Program Counter. If 's' = 1, the contents of the shadow registers, WS, STATUS and BSRS, are loaded into their corresponding registers, W, Status and BSR. If 's' = 0, no update of these registers occurs (default). |                        |      |              |      |                   |      |      |  |
| Words:            | 1                                                                                                                                                                                                                                                                                                                  |                        |      |              |      |                   |      |      |  |
| Cycles:           | 2                                                                                                                                                                                                                                                                                                                  |                        |      |              |      |                   |      |      |  |
| Q Cycle Activity: |                                                                                                                                                                                                                                                                                                                    |                        |      |              |      |                   |      |      |  |
| Q1                |                                                                                                                                                                                                                                                                                                                    | Q2                     |      | Q3           |      | Q4                |      |      |  |
| Decode            |                                                                                                                                                                                                                                                                                                                    | No operation           |      | Process Data |      | POP PC from stack |      |      |  |
| No operation      |                                                                                                                                                                                                                                                                                                                    | No operation           |      | No operation |      | No operation      |      |      |  |

**Example:** RETURN  
After Instruction:  
PC = TOS

| RLCF              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | Rotate Left f through Carry |                      |  |  |      |      |      |      |        |                   |              |                      |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------|----------------------|--|--|------|------|------|------|--------|-------------------|--------------|----------------------|
| Syntax:           | RLCF f {,d {,a}}                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                             |                      |  |  |      |      |      |      |        |                   |              |                      |
| Operands:         | 0 ≤ f ≤ 255<br>d ∈ [0,1]<br>a ∈ [0,1]                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                             |                      |  |  |      |      |      |      |        |                   |              |                      |
| Operation:        | (f<n>) → dest<n + 1>,<br>(f<7>) → C,<br>(C) → dest<0>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                             |                      |  |  |      |      |      |      |        |                   |              |                      |
| Status Affected:  | C, N, Z                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                             |                      |  |  |      |      |      |      |        |                   |              |                      |
| Encoding:         | <table border="1"><tr><td>0011</td><td>01da</td><td>ffff</td><td>ffff</td></tr></table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                             |                      |  |  | 0011 | 01da | ffff | ffff |        |                   |              |                      |
| 0011              | 01da                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | ffff                        | ffff                 |  |  |      |      |      |      |        |                   |              |                      |
| Description:      | <p>The contents of register 'f' are rotated one bit to the left through the CARRY flag. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is stored back in register 'f' (default).</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See <a href="#">Section 42.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</a> for details.</p> <div></div> |                             |                      |  |  |      |      |      |      |        |                   |              |                      |
| Words:            | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                             |                      |  |  |      |      |      |      |        |                   |              |                      |
| Cycles:           | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                             |                      |  |  |      |      |      |      |        |                   |              |                      |
| Q Cycle Activity: | <table><tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr><tr><td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr></table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                             |                      |  |  | Q1   | Q2   | Q3   | Q4   | Decode | Read register 'f' | Process Data | Write to destination |
| Q1                | Q2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Q3                          | Q4                   |  |  |      |      |      |      |        |                   |              |                      |
| Decode            | Read register 'f'                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Process Data                | Write to destination |  |  |      |      |      |      |        |                   |              |                      |

**Example:** RLCF REG, 0, 0

Before Instruction  
REG = 1110 0110  
C = 0  
After Instruction  
REG = 1110 0110  
W = 1100 1100  
C = 1

| RLNCF             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | Rotate Left f (No Carry) |              |                      |      |      |      |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------|--------------|----------------------|------|------|------|
| Syntax:           | RLNCF f {,d {,a}}                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                          |              |                      |      |      |      |
| Operands:         | 0 ≤ f ≤ 255<br>d ∈ [0,1]<br>a ∈ [0,1]                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                          |              |                      |      |      |      |
| Operation:        | (f<n>) → dest<n + 1>,<br>(f<7>) → dest<0>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                          |              |                      |      |      |      |
| Status Affected:  | N, Z                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                          |              |                      |      |      |      |
| Encoding:         | <table><tr><td>0100</td><td>01da</td><td>ffff</td><td>ffff</td></tr></table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                          |              | 0100                 | 01da | ffff | ffff |
| 0100              | 01da                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | ffff                     | ffff         |                      |      |      |      |
| Description:      | <p>The contents of register 'f' are rotated one bit to the left. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is stored back in register 'f' (default). If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See <a href="#">Section 42.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</a> for details.</p> |                          |              |                      |      |      |      |
|                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                          |              |                      |      |      |      |
| Words:            | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                          |              |                      |      |      |      |
| Cycles:           | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                          |              |                      |      |      |      |
| Q Cycle Activity: |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                          |              |                      |      |      |      |
|                   | Q1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | Q2                       | Q3           | Q4                   |      |      |      |
|                   | Decode                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | Read register 'f'        | Process Data | Write to destination |      |      |      |

**Example:** RLNCF REG, 1, 0

Before Instruction

REG = 1010 1011

After Instruction

REG = 0101 0111

| RRCF              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Rotate Right f through Carry |                      |  |  |      |      |      |      |        |                   |              |                      |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------|----------------------|--|--|------|------|------|------|--------|-------------------|--------------|----------------------|
| Syntax:           | RRCF f {,d {,a}}                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                              |                      |  |  |      |      |      |      |        |                   |              |                      |
| Operands:         | 0 ≤ f ≤ 255<br>d ∈ [0,1]<br>a ∈ [0,1]                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                              |                      |  |  |      |      |      |      |        |                   |              |                      |
| Operation:        | (f<n>) → dest<n - 1>,<br>(f<0>) → C,<br>(C) → dest<7>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                              |                      |  |  |      |      |      |      |        |                   |              |                      |
| Status Affected:  | C, N, Z                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                              |                      |  |  |      |      |      |      |        |                   |              |                      |
| Encoding:         | <table><tr><td>0011</td><td>00da</td><td>ffff</td><td>ffff</td></tr></table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                              |                      |  |  | 0011 | 00da | ffff | ffff |        |                   |              |                      |
| 0011              | 00da                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | ffff                         | ffff                 |  |  |      |      |      |      |        |                   |              |                      |
| Description:      | <p>The contents of register 'f' are rotated one bit to the right through the CARRY flag. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See <a href="#">Section 42.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</a> for details.</p> <div></div> |                              |                      |  |  |      |      |      |      |        |                   |              |                      |
| Words:            | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                              |                      |  |  |      |      |      |      |        |                   |              |                      |
| Cycles:           | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                              |                      |  |  |      |      |      |      |        |                   |              |                      |
| Q Cycle Activity: | <table><tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr><tr><td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr></table>                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                              |                      |  |  | Q1   | Q2   | Q3   | Q4   | Decode | Read register 'f' | Process Data | Write to destination |
| Q1                | Q2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Q3                           | Q4                   |  |  |      |      |      |      |        |                   |              |                      |
| Decode            | Read register 'f'                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | Process Data                 | Write to destination |  |  |      |      |      |      |        |                   |              |                      |

**Example:** RRCF REG, 0, 0

Before Instruction

REG = 1110 0110

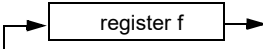
C = 0

After Instruction

REG = 1110 0110

W = 0111 0011

C = 0

| RRNCF             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | Rotate Right f (No Carry) |                      |  |  |      |      |      |      |        |                   |              |                      |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------|----------------------|--|--|------|------|------|------|--------|-------------------|--------------|----------------------|
| Syntax:           | RRNCF f {,d {,a}}                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                           |                      |  |  |      |      |      |      |        |                   |              |                      |
| Operands:         | $0 \leq f \leq 255$<br>$d \in [0,1]$<br>$a \in [0,1]$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                           |                      |  |  |      |      |      |      |        |                   |              |                      |
| Operation:        | $(f<n>) \rightarrow \text{dest}<n-1>$ ,<br>$(f<0>) \rightarrow \text{dest}<7>$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                           |                      |  |  |      |      |      |      |        |                   |              |                      |
| Status Affected:  | N, Z                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                           |                      |  |  |      |      |      |      |        |                   |              |                      |
| Encoding:         | <table><tr><td>0100</td><td>00da</td><td>ffff</td><td>ffff</td></tr></table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                           |                      |  |  | 0100 | 00da | ffff | ffff |        |                   |              |                      |
| 0100              | 00da                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | ffff                      | ffff                 |  |  |      |      |      |      |        |                   |              |                      |
| Description:      | <p>The contents of register 'f' are rotated one bit to the right. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). If 'a' is '0', the Access Bank will be selected (default), overriding the BSR value. If 'a' is '1', then the bank will be selected as per the BSR value. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <a href="#">Section 42.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</a> for details.</p> <div></div> |                           |                      |  |  |      |      |      |      |        |                   |              |                      |
| Words:            | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                           |                      |  |  |      |      |      |      |        |                   |              |                      |
| Cycles:           | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                           |                      |  |  |      |      |      |      |        |                   |              |                      |
| Q Cycle Activity: | <table><tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr><tr><td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr></table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                           |                      |  |  | Q1   | Q2   | Q3   | Q4   | Decode | Read register 'f' | Process Data | Write to destination |
| Q1                | Q2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | Q3                        | Q4                   |  |  |      |      |      |      |        |                   |              |                      |
| Decode            | Read register 'f'                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Process Data              | Write to destination |  |  |      |      |      |      |        |                   |              |                      |

**Example 1:** RRNCF REG, 1, 0

Before Instruction  
 REG = 1101 0111  
 After Instruction  
 REG = 1110 1011

**Example 2:** RRNCF REG, 0, 0

Before Instruction  
 W = ?  
 REG = 1101 0111  
 After Instruction  
 W = 1110 1011  
 REG = 1101 0111

| SETF              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | Set f             |              |                    |  |      |      |      |      |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|--------------|--------------------|--|------|------|------|------|
| Syntax:           | SETF    f {,a}                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                   |              |                    |  |      |      |      |      |
| Operands:         | $0 \leq f \leq 255$<br>$a \in [0,1]$                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                   |              |                    |  |      |      |      |      |
| Operation:        | $\text{FFh} \rightarrow f$                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                   |              |                    |  |      |      |      |      |
| Status Affected:  | None                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                   |              |                    |  |      |      |      |      |
| Encoding:         | <table border="1"><tr><td>0110</td><td>100a</td><td>ffff</td><td>ffff</td></tr></table>                                                                                                                                                                                                                                                                                                                                                                                                  |                   |              |                    |  | 0110 | 100a | ffff | ffff |
| 0110              | 100a                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | ffff              | ffff         |                    |  |      |      |      |      |
| Description:      | <p>The contents of the specified register are set to FFh.</p> <p>If 'a' is '0', the Access Bank is selected.</p> <p>If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <a href="#">Section 42.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</a> for details.</p> |                   |              |                    |  |      |      |      |      |
| Words:            | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                   |              |                    |  |      |      |      |      |
| Cycles:           | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                   |              |                    |  |      |      |      |      |
| Q Cycle Activity: |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                   |              |                    |  |      |      |      |      |
|                   | Q1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | Q2                | Q3           | Q4                 |  |      |      |      |      |
|                   | Decode                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Read register 'f' | Process Data | Write register 'f' |  |      |      |      |      |

**Example:** SETF REG, 1

Before Instruction  
 REG = 5Ah  
 After Instruction  
 REG = FFh

## SLEEP Enter Sleep mode

Syntax: SLEEP

Operands: None

Operation: 00h → WDT,  
0 → WDT postscaler,  
1 →  $\overline{TO}$ ,  
0 →  $\overline{PD}$

Status Affected:  $\overline{TO}$ ,  $\overline{PD}$

Encoding: 

|      |      |      |      |
|------|------|------|------|
| 0000 | 0000 | 0000 | 0011 |
|------|------|------|------|

Description: The Power-down Status bit ( $\overline{PD}$ ) is cleared. The Time-out Status bit ( $\overline{TO}$ ) is set. Watchdog Timer and its postscaler are cleared. The processor is put into Sleep mode with the oscillator stopped.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1     | Q2           | Q3           | Q4          |
|--------|--------------|--------------|-------------|
| Decode | No operation | Process Data | Go to Sleep |

**Example:** SLEEP

Before Instruction

$\overline{TO}$  = ?

$\overline{PD}$  = ?

After Instruction

$\overline{TO}$  = 1 †

$\overline{PD}$  = 0

† If WDT causes wake-up, this bit is cleared.

## SUBFSR Subtract Literal from FSR

Syntax: SUBFSR f, k

Operands:  $0 \leq k \leq 63$   
 $f \in [0, 1, 2]$

Operation:  $FSR(f) - k \rightarrow FSRf$

Status Affected: None

Encoding: 

|      |      |      |      |
|------|------|------|------|
| 1110 | 1001 | ffkk | kkkk |
|------|------|------|------|

Description: The 6-bit literal 'k' is subtracted from the contents of the FSR specified by 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1     | Q2                | Q3           | Q4                   |
|--------|-------------------|--------------|----------------------|
| Decode | Read register 'f' | Process Data | Write to destination |

**Example:** SUBFSR 2, 23h

Before Instruction

FSR2 = 03FFh

After Instruction

FSR2 = 03DCh



## SUBFWB Subtract f from W with borrow

**Syntax:** SUBFWB f {,d {,a}}

**Operands:**  $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

**Operation:**  $(W) - (f) - (\bar{C}) \rightarrow \text{dest}$

**Status Affected:** N, OV, C, DC, Z

**Encoding:**

|      |      |      |      |
|------|------|------|------|
| 0101 | 01da | ffff | ffff |
|------|------|------|------|

**Description:** Subtract register 'f' and CARRY flag (borrow) from W (2's complement method). If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored in register 'f' (default).  
 If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.  
 If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh). See [Section 42.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"](#) for details.

**Words:** 1

**Cycles:** 1

Q Cycle Activity:

| Q1     | Q2                | Q3           | Q4                   |
|--------|-------------------|--------------|----------------------|
| Decode | Read register 'f' | Process Data | Write to destination |

**Example 1:** SUBFWB REG, 1, 0

**Before Instruction**  
 REG = 3  
 W = 2  
 C = 1

**After Instruction**  
 REG = FF  
 W = 2  
 C = 0  
 Z = 0  
 N = 1 ; result is negative

**Example 2:** SUBFWB REG, 0, 0

**Before Instruction**  
 REG = 2  
 W = 5  
 C = 1

**After Instruction**  
 REG = 2  
 W = 3  
 C = 1  
 Z = 0  
 N = 0 ; result is positive

**Example 3:** SUBFWB REG, 1, 0

**Before Instruction**  
 REG = 1  
 W = 2  
 C = 0

**After Instruction**  
 REG = 0  
 W = 2  
 C = 1  
 Z = 1 ; result is zero  
 N = 0

## SUBLW Subtract W from literal

**Syntax:** SUBLW k

**Operands:**  $0 \leq k \leq 255$

**Operation:**  $k - (W) \rightarrow W$

**Status Affected:** N, OV, C, DC, Z

**Encoding:**

|      |      |      |      |
|------|------|------|------|
| 0000 | 1000 | kkkk | kkkk |
|------|------|------|------|

**Description:** W is subtracted from the 8-bit literal 'k'. The result is placed in W.

**Words:** 1

**Cycles:** 1

Q Cycle Activity:

| Q1     | Q2               | Q3           | Q4         |
|--------|------------------|--------------|------------|
| Decode | Read literal 'k' | Process Data | Write to W |

**Example 1:** SUBLW 02h

**Before Instruction**  
 W = 01h  
 C = ?

**After Instruction**  
 W = 01h  
 C = 1 ; result is positive  
 Z = 0  
 N = 0

**Example 2:** SUBLW 02h

**Before Instruction**  
 W = 02h  
 C = ?

**After Instruction**  
 W = 00h  
 C = 1 ; result is zero  
 Z = 1  
 N = 0

**Example 3:** SUBLW 02h

**Before Instruction**  
 W = 03h  
 C = ?

**After Instruction**  
 W = FFh ; (2's complement)  
 C = 0 ; result is negative  
 Z = 0  
 N = 1

## SUBWF Subtract W from f

|                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |      |      |      |
|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|------|------|
| Syntax:          | SUBWF f {,d {,a}}                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |      |      |      |
| Operands:        | $0 \leq f \leq 255$<br>$d \in [0,1]$<br>$a \in [0,1]$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |      |      |      |
| Operation:       | $(f) - (W) \rightarrow \text{dest}$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |      |      |      |
| Status Affected: | N, OV, C, DC, Z                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |      |      |      |
| Encoding:        | 0101                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | 11da | ffff | ffff |
| Description:     | <p>Subtract W from register 'f' (2's complement method). If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default).</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <a href="#">Section 42.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</a> for details.</p> |      |      |      |
| Words:           | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |      |      |      |
| Cycles:          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |      |      |      |

| Q1     | Q2                | Q3           | Q4                   |
|--------|-------------------|--------------|----------------------|
| Decode | Read register 'f' | Process Data | Write to destination |

**Example 1:** SUBWF REG, 1, 0

Before Instruction

|     |   |   |
|-----|---|---|
| REG | = | 3 |
| W   | = | 2 |
| C   | = | ? |

After Instruction

|     |   |                        |
|-----|---|------------------------|
| REG | = | 1                      |
| W   | = | 2                      |
| C   | = | 1 ; result is positive |
| Z   | = | 0                      |
| N   | = | 0                      |

**Example 2:** SUBWF REG, 0, 0

Before Instruction

|     |   |   |
|-----|---|---|
| REG | = | 2 |
| W   | = | 2 |
| C   | = | ? |

After Instruction

|     |   |                    |
|-----|---|--------------------|
| REG | = | 2                  |
| W   | = | 0                  |
| C   | = | 1 ; result is zero |
| Z   | = | 1                  |
| N   | = | 0                  |

**Example 3:** SUBWF REG, 1, 0

Before Instruction

|     |   |   |
|-----|---|---|
| REG | = | 1 |
| W   | = | 2 |
| C   | = | ? |

After Instruction

|     |   |                        |
|-----|---|------------------------|
| REG | = | FFh ;(2's complement)  |
| W   | = | 2                      |
| C   | = | 0 ; result is negative |
| Z   | = | 0                      |
| N   | = | 1                      |

## SUBWFB Subtract W from f with Borrow

|                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |      |      |  |      |      |      |      |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|------|--|------|------|------|------|
| Syntax:           | SUBWFB f {,d {,a}}                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |      |      |  |      |      |      |      |
| Operands:         | $0 \leq f \leq 255$<br>$d \in [0,1]$<br>$a \in [0,1]$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |      |      |  |      |      |      |      |
| Operation:        | $(f) - (W) - (\overline{C}) \rightarrow \text{dest}$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |      |      |  |      |      |      |      |
| Status Affected:  | N, OV, C, DC, Z                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |      |      |  |      |      |      |      |
| Encoding:         | <table border="1"><tr><td>0101</td><td>10da</td><td>ffff</td><td>ffff</td></tr></table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |      |      |  | 0101 | 10da | ffff | ffff |
| 0101              | 10da                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | ffff | ffff |  |      |      |      |      |
| Description:      | <p>Subtract W and the CARRY flag (borrow) from register 'f' (2's complement method). If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default). If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <a href="#">Section 42.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</a> for details.</p> |      |      |  |      |      |      |      |
| Words:            | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |      |      |  |      |      |      |      |
| Cycles:           | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |      |      |  |      |      |      |      |
| Q Cycle Activity: |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |      |      |  |      |      |      |      |

| Q1     | Q2                | Q3           | Q4                   |
|--------|-------------------|--------------|----------------------|
| Decode | Read register 'f' | Process Data | Write to destination |

**Example 1:** SUBWFB REG, 1, 0

Before Instruction

|     |   |     |             |
|-----|---|-----|-------------|
| REG | = | 19h | (0001 1001) |
| W   | = | 0Dh | (0000 1101) |
| C   | = | 1   |             |

After Instruction

|     |   |     |                      |
|-----|---|-----|----------------------|
| REG | = | 0Ch | (0000 1100)          |
| W   | = | 0Dh | (0000 1101)          |
| C   | = | 1   |                      |
| Z   | = | 0   |                      |
| N   | = | 0   | ; result is positive |

**Example 2:** SUBWFB REG, 0, 0

Before Instruction

|     |   |     |             |
|-----|---|-----|-------------|
| REG | = | 1Bh | (0001 1011) |
| W   | = | 1Ah | (0001 1010) |
| C   | = | 0   |             |

After Instruction

|     |   |     |                  |
|-----|---|-----|------------------|
| REG | = | 1Bh | (0001 1011)      |
| W   | = | 00h |                  |
| C   | = | 1   |                  |
| Z   | = | 1   | ; result is zero |
| N   | = | 0   |                  |

**Example 3:** SUBWFB REG, 1, 0

Before Instruction

|     |   |     |             |
|-----|---|-----|-------------|
| REG | = | 03h | (0000 0011) |
| W   | = | 0Eh | (0000 1110) |
| C   | = | 1   |             |

After Instruction

|     |   |     |                             |
|-----|---|-----|-----------------------------|
| REG | = | F5h | (1111 0101)<br>; [2's comp] |
| W   | = | 0Eh | (0000 1110)                 |
| C   | = | 0   |                             |
| Z   | = | 0   |                             |
| N   | = | 1   | ; result is negative        |

## SWAPF

## Swap f

Syntax: SWAPF f {,d {,a}}

Operands:  $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

Operation:  $(f<3:0>) \rightarrow \text{dest}<7:4>$ ,  
 $(f<7:4>) \rightarrow \text{dest}<3:0>$

Status Affected: None

Encoding: 

|      |      |      |      |
|------|------|------|------|
| 0011 | 10da | ffff | ffff |
|------|------|------|------|

Description: The upper and lower nibbles of register 'f' are exchanged. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed in register 'f' (default). If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh). See [Section 42.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"](#) for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1     | Q2                | Q3           | Q4                   |
|--------|-------------------|--------------|----------------------|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: SWAPF REG, 1, 0

Before Instruction

REG = 53h

After Instruction

REG = 35h

## TBLRD Table Read

**Syntax:** TBLRD ( \*,\*+,\*-,+\*)

**Operands:** None

**Operation:** if TBLRD \*,  
(Prog Mem (TBLPTR)) → TABLAT;  
TBLPTR – No Change;  
if TBLRD \*+,  
(Prog Mem (TBLPTR)) → TABLAT;  
(TBLPTR) + 1 → TBLPTR;  
if TBLRD \*-,  
(Prog Mem (TBLPTR)) → TABLAT;  
(TBLPTR) – 1 → TBLPTR;  
if TBLRD +\*,  
(TBLPTR) + 1 → TBLPTR;  
(Prog Mem (TBLPTR)) → TABLAT;

**Status Affected:** None

|           |      |      |      |                                           |
|-----------|------|------|------|-------------------------------------------|
| Encoding: | 0000 | 0000 | 0000 | 10nn<br>nn=0 *<br>=1 *+<br>=2 *-<br>=3 +* |
|-----------|------|------|------|-------------------------------------------|

**Description:** This instruction is used to read the contents of Program Memory (P.M.). To address the program memory, a pointer called Table Pointer (TBLPTR) is used. The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2-Mbyte address range.

TBLPTR[0] = 0: Least Significant Byte of Program Memory Word  
TBLPTR[0] = 1: Most Significant Byte of Program Memory Word

The TBLRD instruction can modify the value of TBLPTR as follows:

- no change
- post-increment
- post-decrement
- pre-increment

**Words:** 1

**Cycles:** 2

**Q Cycle Activity:**

| Q1           | Q2                                 | Q3           | Q4                          |
|--------------|------------------------------------|--------------|-----------------------------|
| Decode       | No operation                       | No operation | No operation                |
| No operation | No operation (Read Program Memory) | No operation | No operation (Write TABLAT) |

## TBLRD Table Read (Continued)

**Example1:** TBLRD \*+ ;

Before Instruction

|                  |   |         |
|------------------|---|---------|
| TABLAT           | = | 55h     |
| TBLPTR           | = | 00A356h |
| MEMORY (00A356h) | = | 34h     |

After Instruction

|        |   |         |
|--------|---|---------|
| TABLAT | = | 34h     |
| TBLPTR | = | 00A357h |

**Example2:** TBLRD +- ;

Before Instruction

|                  |   |         |
|------------------|---|---------|
| TABLAT           | = | AAh     |
| TBLPTR           | = | 01A357h |
| MEMORY (01A357h) | = | 12h     |
| MEMORY (01A358h) | = | 34h     |

After Instruction

|        |   |         |
|--------|---|---------|
| TABLAT | = | 34h     |
| TBLPTR | = | 01A358h |

| TBLWT             | Table Write                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |      |                                           |      |                                           |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|-------------------------------------------|------|-------------------------------------------|
| Syntax:           | TBLWT ( *, *+; *-, +* )                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |      |                                           |      |                                           |
| Operands:         | None                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |      |                                           |      |                                           |
| Operation:        | if TBLWT*,<br>(TABLAT) → Holding Register;<br>TBLPTR – No Change;<br>if TBLWT*+,<br>(TABLAT) → Holding Register;<br>(TBLPTR) + 1 → TBLPTR;<br>if TBLWT*-,<br>(TABLAT) → Holding Register;<br>(TBLPTR) – 1 → TBLPTR;<br>if TBLWT+*,<br>(TBLPTR) + 1 → TBLPTR;<br>(TABLAT) → Holding Register;                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |      |                                           |      |                                           |
| Status Affected:  | None                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |      |                                           |      |                                           |
| Encoding:         | <table><tr><td>0000</td><td>0000</td><td>0000</td><td>11nn<br/>nn=0 *<br/>=1 *+<br/>=2 *-<br/>=3 +*</td></tr></table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | 0000 | 0000                                      | 0000 | 11nn<br>nn=0 *<br>=1 *+<br>=2 *-<br>=3 +* |
| 0000              | 0000                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | 0000 | 11nn<br>nn=0 *<br>=1 *+<br>=2 *-<br>=3 +* |      |                                           |
| Description:      | <p>This instruction uses the three LSBs of TBLPTR to determine which of the eight holding registers the TABLAT is written to. The holding registers are used to program the contents of Program Memory (P.M.). (Refer to <a href="#">Section 13.1 “Program Flash Memory”</a> for additional details on programming Flash memory.)</p> <p>The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2-MByte address range. The LSB of the TBLPTR selects which byte of the program memory location to access.</p> <p>TBLPTR[0] = 0: Least Significant Byte of Program Memory Word</p> <p>TBLPTR[0] = 1: Most Significant Byte of Program Memory Word</p> <p>The TBLWT instruction can modify the value of TBLPTR as follows:</p> <ul style="list-style-type: none"><li>• no change</li><li>• post-increment</li><li>• post-decrement</li><li>• pre-increment</li></ul> |      |                                           |      |                                           |
| Words:            | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |      |                                           |      |                                           |
| Cycles:           | 2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |      |                                           |      |                                           |
| Q Cycle Activity: |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |      |                                           |      |                                           |

|              | Q1                         | Q2           | Q3           | Q4                                       |
|--------------|----------------------------|--------------|--------------|------------------------------------------|
| Decode       | No operation               | No operation | No operation | No operation                             |
| No operation | No operation (Read TABLAT) | No operation | No operation | No operation (Write to Holding Register) |

| TBLWT                                       | Table Write (Continued) |
|---------------------------------------------|-------------------------|
| <u>Example 1:</u>                           | TBLWT *+;               |
| Before Instruction                          |                         |
| TABLAT                                      | = 55h                   |
| TBLPTR                                      | = 00A356h               |
| HOLDING REGISTER (00A356h)                  | = FFh                   |
| After Instructions (table write completion) |                         |
| TABLAT                                      | = 55h                   |
| TBLPTR                                      | = 00A357h               |
| HOLDING REGISTER (00A356h)                  | = 55h                   |
| <u>Example 2:</u>                           | TBLWT *+;               |
| Before Instruction                          |                         |
| TABLAT                                      | = 34h                   |
| TBLPTR                                      | = 01389Ah               |
| HOLDING REGISTER (01389Ah)                  | = FFh                   |
| HOLDING REGISTER (01389Bh)                  | = FFh                   |
| After Instruction (table write completion)  |                         |
| TABLAT                                      | = 34h                   |
| TBLPTR                                      | = 01389Bh               |
| HOLDING REGISTER (01389Ah)                  | = FFh                   |
| HOLDING REGISTER (01389Bh)                  | = 34h                   |

## TSTFSZ Test f, skip if 0

|                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |      |      |      |      |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|------|------|------|
| Syntax:          | TSTFSZ f {,a}                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |      |      |      |      |
| Operands:        | $0 \leq f \leq 255$<br>$a \in [0,1]$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |      |      |      |      |
| Operation:       | skip if $f = 0$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |      |      |      |      |
| Status Affected: | None                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |      |      |      |      |
| Encoding:        | <table border="1"><tr><td>0110</td><td>011a</td><td>ffff</td><td>ffff</td></tr></table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | 0110 | 011a | ffff | ffff |
| 0110             | 011a                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | ffff | ffff |      |      |
| Description:     | <p>If 'f' = 0, the next instruction fetched during the current instruction execution is discarded and a NOP is executed, making this a 2-cycle instruction.</p> <p>If 'a' is '0', the Access Bank is selected.</p> <p>If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <a href="#">Section 42.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</a> for details.</p> |      |      |      |      |
| Words:           | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |      |      |      |      |
| Cycles:          | 1(2)<br><b>Note:</b> 3 cycles if skip and followed by a 2-word instruction.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |      |      |      |      |

Q Cycle Activity:

| Q1     | Q2                | Q3           | Q4           |
|--------|-------------------|--------------|--------------|
| Decode | Read register 'f' | Process Data | No operation |

If skip:

| Q1           | Q2           | Q3           | Q4           |
|--------------|--------------|--------------|--------------|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1           | Q2           | Q3           | Q4           |
|--------------|--------------|--------------|--------------|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

**Example:**

```

HERE    TSTFSZ  CNT, 1
NZERO   :
ZERO    :
```

Before Instruction

PC = Address (HERE)

After Instruction

```

If CNT = 00h,
PC = Address (ZERO)
If CNT ≠ 00h,
PC = Address (NZERO)
```

## XORLW Exclusive OR literal with W

|                  |                                                                                         |      |      |      |      |
|------------------|-----------------------------------------------------------------------------------------|------|------|------|------|
| Syntax:          | XORLW k                                                                                 |      |      |      |      |
| Operands:        | $0 \leq k \leq 255$                                                                     |      |      |      |      |
| Operation:       | (W) .XOR. $k \rightarrow W$                                                             |      |      |      |      |
| Status Affected: | N, Z                                                                                    |      |      |      |      |
| Encoding:        | <table border="1"><tr><td>0000</td><td>1010</td><td>kkkk</td><td>kkkk</td></tr></table> | 0000 | 1010 | kkkk | kkkk |
| 0000             | 1010                                                                                    | kkkk | kkkk |      |      |
| Description:     | The contents of W are XORed with the 8-bit literal 'k'. The result is placed in W.      |      |      |      |      |
| Words:           | 1                                                                                       |      |      |      |      |
| Cycles:          | 1                                                                                       |      |      |      |      |

Q Cycle Activity:

| Q1     | Q2               | Q3           | Q4         |
|--------|------------------|--------------|------------|
| Decode | Read literal 'k' | Process Data | Write to W |

**Example:** XORLW 0AFh

Before Instruction

W = B5h

After Instruction

W = 1Ah

| XORWF             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | Exclusive OR W with f |              |                      |  |      |      |      |      |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|--------------|----------------------|--|------|------|------|------|
| Syntax:           | XORWF    f {,d {,a}}                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                       |              |                      |  |      |      |      |      |
| Operands:         | 0 ≤ f ≤ 255<br>d ∈ [0,1]<br>a ∈ [0,1]                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                       |              |                      |  |      |      |      |      |
| Operation:        | (W) .XOR. (f) → dest                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                       |              |                      |  |      |      |      |      |
| Status Affected:  | N, Z                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                       |              |                      |  |      |      |      |      |
| Encoding:         | <table border="1"><tr><td>0001</td><td>10da</td><td>ffff</td><td>ffff</td></tr></table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                       |              |                      |  | 0001 | 10da | ffff | ffff |
| 0001              | 10da                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | ffff                  | ffff         |                      |  |      |      |      |      |
| Description:      | <p>Exclusive OR the contents of W with register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in the register 'f' (default).</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See <a href="#">Section 42.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</a> for details.</p> |                       |              |                      |  |      |      |      |      |
| Words:            | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                       |              |                      |  |      |      |      |      |
| Cycles:           | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                       |              |                      |  |      |      |      |      |
| Q Cycle Activity: |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                       |              |                      |  |      |      |      |      |
|                   | Q1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Q2                    | Q3           | Q4                   |  |      |      |      |      |
|                   | Decode                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Read register 'f'     | Process Data | Write to destination |  |      |      |      |      |

**Example:** XORWF REG, 1, 0

Before Instruction

REG = AFh

W = B5h

After Instruction

REG = 1Ah

W = B5h

## 42.2 Extended Instruction Set

In addition to the standard 75 instructions of the PIC18 instruction set, PIC18(L)F25/26K83 devices also provide an optional extension to the core CPU functionality. The added features include eight additional instructions that augment indirect and indexed addressing operations and the implementation of Indexed Literal Offset Addressing mode for many of the standard PIC18 instructions.

The additional features of the extended instruction set are disabled by default. To enable them, users must set the XINST Configuration bit.

The instructions in the extended set can all be classified as literal operations, which either manipulate the File Select Registers, or use them for indexed addressing. Two of the instructions, `ADDFSR` and `SUBFSR`, each have an additional special instantiation for using FSR2. These versions (`ADDULNK` and `SUBULNK`) allow for automatic return after execution.

The extended instructions are specifically implemented to optimize re-entrant program code (that is, code that is recursive or that uses a software stack) written in high-level languages, particularly C. Among other things, they allow users working in high-level languages to perform certain operations on data structures more efficiently. These include:

- dynamic allocation and deallocation of software stack space when entering and leaving subroutines
- function pointer invocation
- software Stack Pointer manipulation
- manipulation of variables located in a software stack

A summary of the instructions in the extended instruction set is provided in [Table 42-3](#). Detailed descriptions are provided in [Section 42.2.2 “Extended Instruction Set”](#). The opcode field descriptions in [Table 42-1](#) apply to both the standard and extended PIC18 instruction sets.

**Note:** The instruction set extension and the Indexed Literal Offset Addressing mode were designed for optimizing applications written in C; the user may likely never use these instructions directly in assembler. The syntax for these commands is provided as a reference for users who may be reviewing code that has been generated by a compiler.

### 42.2.1 EXTENDED INSTRUCTION SYNTAX

Most of the extended instructions use indexed arguments, using one of the File Select Registers and some offset to specify a source or destination register. When an argument for an instruction serves as part of indexed addressing, it is enclosed in square brackets (“[]”). This is done to indicate that the argument is used as an index or offset. MPASM™ Assembler will flag an error if it determines that an index or offset value is not bracketed.

When the extended instruction set is enabled, brackets are also used to indicate index arguments in byte-oriented and bit-oriented instructions. This is in addition to other changes in their syntax. For more details, see [Section 42.2.3.1 “Extended Instruction Syntax with Standard PIC18 Commands”](#).

**Note:** In the past, square brackets have been used to denote optional arguments in the PIC18 and earlier instruction sets. In this text and going forward, optional arguments are denoted by braces (“{ }”).



**TABLE 42-3: EXTENSIONS TO THE PIC18 INSTRUCTION SET**

| Mnemonic,<br>Operands |                                 | Description                                | Cycles | 16-Bit Instruction Word |      |      |      | Status<br>Affected |
|-----------------------|---------------------------------|--------------------------------------------|--------|-------------------------|------|------|------|--------------------|
|                       |                                 |                                            |        | MSb                     |      | LSb  |      |                    |
| ADDULNK               | k                               | Add FSR2 with (k) & return                 | 2      | 1110                    | 1000 | 11kk | kkkk | None               |
| MOVSF                 | z <sub>s</sub> , f <sub>d</sub> | Move z <sub>s</sub> (source) to 1st word   | 2      | 1110                    | 1011 | 0zzz | zzzz | None               |
|                       |                                 | f <sub>d</sub> (destination) 2nd word      | 2      | 1111                    | ffff | ffff | ffff |                    |
| MOVSFL                | z <sub>s</sub> , f <sub>d</sub> | Opcode 1st word                            |        | 0000                    | 0000 | 0000 | 0010 | None               |
|                       |                                 | Move z <sub>s</sub> (source) to 2nd word   | 3      | 1111                    | xxxz | zzzz | zzff |                    |
|                       |                                 | f <sub>d</sub> (full destination) 3rd word |        | 1111                    | ffff | ffff | ffff |                    |
| MOVSS                 | z <sub>s</sub> , z <sub>d</sub> | Move z <sub>s</sub> (source) to 1st word   |        | 1110                    | 1011 | 1zzz | zzzz | None               |
|                       |                                 | z <sub>d</sub> (destination) 2nd word      | 2      | 1111                    | xxxx | xzzz | zzzz |                    |
| PUSHL                 | k                               | Push literal to POSTDEC2                   | 1      | 1110                    | 1010 | kkkk | kkkk | None               |
| SUBULNK               | k                               | Subtract (k) from FSR2 & return            | 2      | 1110                    | 1001 | 11kk | kkkk | None               |

- Note 1:** If Program Counter (PC) is modified or a conditional test is true, the instruction requires an additional cycle. The extra cycle is executed as a NOP.
- 2:** Some instructions are multi word instructions. The second/third words of these instructions will be decoded as a NOP, unless the first word of the instruction retrieves the information embedded in these 16-bits. This ensures that all program memory locations have a valid instruction.
- 3:** Only available when extended instruction set is enabled.
- 4:**  $f_s$  and  $f_d$  do not cover the full memory range. Two MSBs of bank selection are forced to 'b00 to limit the range of these instructions to lower 4k addressing space.

## 42.2.2 EXTENDED INSTRUCTION SET

### **ADDULNK      Add Literal to FSR2 and Return**

|                  |                                                                                                                                                                                                                                                                                                                                                                               |      |      |      |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|------|------|
| Syntax:          | ADDULNK k                                                                                                                                                                                                                                                                                                                                                                     |      |      |      |
| Operands:        | $0 \leq k \leq 63$                                                                                                                                                                                                                                                                                                                                                            |      |      |      |
| Operation:       | FSR2 + k → FSR2,<br>(TOS) → PC                                                                                                                                                                                                                                                                                                                                                |      |      |      |
| Status Affected: | None                                                                                                                                                                                                                                                                                                                                                                          |      |      |      |
| Encoding:        | 1110                                                                                                                                                                                                                                                                                                                                                                          | 1000 | 11kk | kkkk |
| Description:     | <p>The 6-bit literal 'k' is added to the contents of FSR2. A <b>RETURN</b> is then executed by loading the PC with the TOS.</p> <p>The instruction takes two cycles to execute; a <b>NOP</b> is performed during the second cycle.</p> <p>This may be thought of as a special case of the <b>ADDFSR</b> instruction, where f = 3 (binary '11'); it operates only on FSR2.</p> |      |      |      |
| Words:           | 1                                                                                                                                                                                                                                                                                                                                                                             |      |      |      |
| Cycles:          | 2                                                                                                                                                                                                                                                                                                                                                                             |      |      |      |

Q Cycle Activity:

| Q1              | Q2                  | Q3              | Q4              |
|-----------------|---------------------|-----------------|-----------------|
| Decode          | Read<br>literal 'k' | Process<br>Data | Write to<br>FSR |
| No<br>Operation | No<br>Operation     | No<br>Operation | No<br>Operation |

**Example:**      ADDULNK 23h

Before Instruction

FSR2 = 03FFh  
PC = 0100h

After Instruction

FSR2 = 0422h  
PC = (TOS)

**Note:** All PIC18 instructions may take an optional label argument preceding the instruction mnemonic for use in symbolic addressing. If a label is used, the instruction syntax then becomes: {label} instruction argument(s).

## MOVSF Move Indexed to f

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax:            | MOVSF [z <sub>s</sub> ], f <sub>d</sub>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Operands:          | 0 ≤ z <sub>s</sub> ≤ 127<br>0 ≤ f <sub>d</sub> ≤ 4095                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Operation:         | ((FSR2) + z <sub>s</sub> ) → f <sub>d</sub>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| Status Affected:   | None                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Encoding:          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 1st word (source)  | 1110 1011 0zzz zzzz <sub>s</sub>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 2nd word (destin.) | 1111 ffff ffff ffff <sub>d</sub>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Description:       | The contents of the source register are moved to destination register 'f <sub>d</sub> '. The actual address of the source register is determined by adding the 7-bit literal offset 'z <sub>s</sub> ' in the first word to the value of FSR2. The address of the destination register is specified by the 12-bit literal 'f <sub>d</sub> ' in the second word. Both addresses can be anywhere in the 4096-byte data space (000h to FFFh).<br>MOVSF has curtailed the destination range to the lower 4 Kbyte space in memory (Banks 1 through 15). For everything else, use MOVSFL. |
| Words:             | 2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Cycles:            | 2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Q Cycle Activity:  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

| Q1     | Q2                            | Q3                    | Q4                        |
|--------|-------------------------------|-----------------------|---------------------------|
| Decode | Determine source addr         | Determine source addr | Read source reg           |
| Decode | No operation<br>No dummy read | No operation          | Write register 'f' (dest) |

**Example:** MOVSF [05h], REG2

|                    |       |
|--------------------|-------|
| Before Instruction |       |
| FSR2               | = 80h |
| Contents of 85h    | = 33h |
| REG2               | = 11h |
| After Instruction  |       |
| FSR2               | = 80h |
| Contents of 85h    | = 33h |
| REG2               | = 33h |

## MOVSFL Move Indexed to f (Long Range)

|                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax:                 | MOVSFL [z <sub>s</sub> ], f <sub>d</sub>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Operands:               | 0 ≤ z <sub>s</sub> ≤ 127<br>0 ≤ f <sub>d</sub> ≤ 16383                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| Operation:              | ((FSR2) + z <sub>s</sub> ) → f <sub>d</sub>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Status Affected:        | None                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Encoding:               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 1st word (opcode)       | 0000 0000 0110 0010                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 2nd word (source)       | 1111 xxxx zzzz zzzz <sub>s</sub> ff                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 3rd word (full destin.) | 1111 ffff ffff ffff <sub>d</sub>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Description:            | The contents of the source register are moved to destination register 'f <sub>d</sub> '. The actual address of the source register is determined by adding the 7-bit literal offset 'z <sub>s</sub> ' in the first word to the value of FSR2 (14 bits). The address of the destination register is specified by the 14-bit literal 'f <sub>d</sub> ' in the second word. Both addresses can be anywhere in the 16 Kbyte data space (0000h to 3FFFh). The MOVSFL instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register. If the resultant source address points to an indirect addressing register, the value returned will be 00h. |
| Words:                  | 3                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| Cycles:                 | 3                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| Q Cycle Activity:       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |

| Q1     | Q2                            | Q3           | Q4                         |
|--------|-------------------------------|--------------|----------------------------|
| Decode | No operation                  | No operation | No operation               |
| Decode | Read register "z" (src.)      | Process data | No operation               |
| Decode | No operation<br>No dummy read | No operation | Write register "f" (dest.) |

**Example:** MOVSFL [05h], REG2

|                    |       |
|--------------------|-------|
| Before Instruction |       |
| FSR2               | = 80h |
| Contents of 85h    | = 33h |
| REG2               | = 11h |
| After Instruction  |       |
| FSR2               | = 80h |
| Contents of 85h    | = 33h |

## MOVSS Move Indexed to Indexed

**Syntax:** MOVSS [z<sub>s</sub>], [z<sub>d</sub>]

**Operands:** 0 ≤ z<sub>s</sub> ≤ 127  
0 ≤ z<sub>d</sub> ≤ 127

**Operation:** ((FSR2) + z<sub>s</sub>) → ((FSR2) + z<sub>d</sub>)

**Status Affected:** None

**Encoding:**

|      |      |      |                   |
|------|------|------|-------------------|
| 1110 | 1011 | 1zzz | zzzz <sub>s</sub> |
| 1111 | xxxx | xzzz | zzzz <sub>d</sub> |

**1st word (source)**

**2nd word (dest.)**

**Description** The contents of the source register are moved to the destination register. The addresses of the source and destination registers are determined by adding the 7-bit literal offsets 'z<sub>s</sub>' or 'z<sub>d</sub>', respectively, to the value of FSR2. Both registers can be located anywhere in the 4096-byte data memory space (000h to FFFh). The MOVSS instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register. If the resultant source address points to an indirect addressing register, the value returned will be 00h. If the resultant destination address points to an indirect addressing register, the instruction will execute as a NOP.

**Words:** 2

**Cycles:** 2

**Q Cycle Activity:**

| Q1     | Q2                    | Q3                    | Q4                |
|--------|-----------------------|-----------------------|-------------------|
| Decode | Determine source addr | Determine source addr | Read source reg   |
| Decode | Determine dest addr   | Determine dest addr   | Write to dest reg |

**Example:** MOVSS [05h], [06h]

**Before Instruction**

FSR2 = 80h

Contents of 85h = 33h

Contents of 86h = 11h

**After Instruction**

FSR2 = 80h

Contents of 85h = 33h

Contents of 86h = 33h

## PUSHL Store Literal at FSR2, Decrement FSR2

**Syntax:** PUSHL k

**Operands:** 0 ≤ k ≤ 255

**Operation:** k → (FSR2),  
FSR2 – 1 → FSR2

**Status Affected:** None

**Encoding:**

|      |      |      |      |
|------|------|------|------|
| 1111 | 1010 | kkkk | kkkk |
|------|------|------|------|

**Description:** The 8-bit literal 'k' is written to the data memory address specified by FSR2. FSR2 is decremented by 1 after the operation. This instruction allows users to push values onto a software stack.

**Words:** 1

**Cycles:** 1

**Q Cycle Activity:**

| Q1     | Q2       | Q3           | Q4                   |
|--------|----------|--------------|----------------------|
| Decode | Read 'k' | Process data | Write to destination |

**Example:** PUSHL 08h

**Before Instruction**

FSR2H:FSR2L = 01ECh

Memory (01ECh) = 00h

**After Instruction**

FSR2H:FSR2L = 01EBh

Memory (01ECh) = 08h

## **SUBULNK     Subtract Literal from FSR2 and Return**

Syntax:            SUBULNK   k

Operands:         $0 \leq k \leq 63$

Operation:        FSR2 – k → FSR2  
(TOS) → PC

Status Affected: None

Encoding:        

|      |      |      |      |
|------|------|------|------|
| 1110 | 1001 | 11kk | kkkk |
|------|------|------|------|

Description:     The 6-bit literal 'k' is subtracted from the contents of the FSR2. A RETURN is then executed by loading the PC with the TOS. The instruction takes two cycles to execute; a NOP is performed during the second cycle. This may be thought of as a special case of the SUBFSR instruction, where f = 3 (binary '11'); it operates only on FSR2.

Words:            1

Cycles:           2

Q Cycle Activity:

| Q1           | Q2                | Q3           | Q4                   |
|--------------|-------------------|--------------|----------------------|
| Decode       | Read register 'f' | Process Data | Write to destination |
| No Operation | No Operation      | No Operation | No Operation         |

Example:                SUBULNK 23h

Before Instruction

FSR2    =    03FFh

PC        =    0100h

After Instruction

FSR2    =    03DCh

PC        =    (TOS)

## 42.2.3 BYTE-ORIENTED AND BIT-ORIENTED INSTRUCTIONS IN INDEXED LITERAL OFFSET MODE

**Note:** Enabling the PIC18 instruction set extension may cause legacy applications to behave erratically or fail entirely.

In addition to eight new commands in the extended set, enabling the extended instruction set also enables Indexed Literal Offset Addressing mode ([Section 4.8.1 “Indexed Addressing with Literal Offset”](#)). This has a significant impact on the way that many commands of the standard PIC18 instruction set are interpreted.

When the extended set is disabled, addresses embedded in opcodes are treated as literal memory locations: either as a location in the Access Bank ('a' = 0), or in a GPR bank designated by the BSR ('a' = 1). When the extended instruction set is enabled and 'a' = 0, however, a file register argument of 5Fh or less is interpreted as an offset from the pointer value in FSR2 and not as a literal address. For practical purposes, this means that all instructions that use the Access RAM bit as an argument – that is, all byte-oriented and bit-oriented instructions, or almost half of the core PIC18 instructions – may behave differently when the extended instruction set is enabled.

When the content of FSR2 is 00h, the boundaries of the Access RAM are essentially remapped to their original values. This may be useful in creating backward compatible code. If this technique is used, it may be necessary to save the value of FSR2 and restore it when moving back and forth between C and assembly routines in order to preserve the Stack Pointer. Users must also keep in mind the syntax requirements of the extended instruction set (see [Section 42.2.3.1 “Extended Instruction Syntax with Standard PIC18 Commands”](#)).

Although the Indexed Literal Offset Addressing mode can be very useful for dynamic stack and pointer manipulation, it can also be very annoying if a simple arithmetic operation is carried out on the wrong register. Users who are accustomed to the PIC18 programming must keep in mind that, when the extended instruction set is enabled, register addresses of 5Fh or less are used for Indexed Literal Offset Addressing.

Representative examples of typical byte-oriented and bit-oriented instructions in the Indexed Literal Offset Addressing mode are provided on the following page to show how execution is affected. The operand conditions shown in the examples are applicable to all instructions of these types.

### 42.2.3.1 Extended Instruction Syntax with Standard PIC18 Commands

When the extended instruction set is enabled, the file register argument, 'f', in the standard byte-oriented and bit-oriented commands is replaced with the literal offset value, 'k'. As already noted, this occurs only when 'f' is less than or equal to 5Fh. When an offset value is used, it must be indicated by square brackets ("[]"). As with the extended instructions, the use of brackets indicates to the compiler that the value is to be interpreted as an index or an offset. Omitting the brackets, or using a value greater than 5Fh within brackets, will generate an error in the MPASM assembler.

If the index argument is properly bracketed for Indexed Literal Offset Addressing, the Access RAM argument is never specified; it will automatically be assumed to be '0'. This is in contrast to standard operation (extended instruction set disabled) when 'a' is set on the basis of the target address. Declaring the Access RAM bit in this mode will also generate an error in the MPASM assembler.

The destination argument, 'd', functions as before.

In the latest versions of the MPASM™ assembler, language support for the extended instruction set must be explicitly invoked. This is done with either the command line option, /y, or the PE directive in the source listing.

## 42.2.4 CONSIDERATIONS WHEN ENABLING THE EXTENDED INSTRUCTION SET

It is important to note that the extensions to the instruction set may not be beneficial to all users. In particular, users who are not writing code that uses a software stack may not benefit from using the extensions to the instruction set.

Additionally, the Indexed Literal Offset Addressing mode may create issues with legacy applications written to the PIC18 assembler. This is because instructions in the legacy code may attempt to address registers in the Access Bank below 5Fh. Since these addresses are interpreted as literal offsets to FSR2 when the instruction set extension is enabled, the application may read or write to the wrong data addresses.

When porting an application to the PIC18(L)F25/26K83, it is very important to consider the type of code. A large, re-entrant application that is written in 'C' and would benefit from efficient compilation will do well when using the instruction set extensions. Legacy applications that heavily use the Access Bank will most likely not benefit from using the extended instruction set.

| ADDWF             |                                                                                                                                                                                                                          | ADD W to Indexed<br>(Indexed Literal Offset mode) |              |                      |      |      |      |      |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------|--------------|----------------------|------|------|------|------|
| Syntax:           | ADDWF [k] {,d}                                                                                                                                                                                                           |                                                   |              |                      |      |      |      |      |
| Operands:         | $0 \leq k \leq 95$<br>$d \in [0,1]$                                                                                                                                                                                      |                                                   |              |                      |      |      |      |      |
| Operation:        | $(W) + ((FSR2) + k) \rightarrow dest$                                                                                                                                                                                    |                                                   |              |                      |      |      |      |      |
| Status Affected:  | N, OV, C, DC, Z                                                                                                                                                                                                          |                                                   |              |                      |      |      |      |      |
| Encoding:         | <table border="1"><tr><td>0010</td><td>01d0</td><td>kkkk</td><td>kkkk</td></tr></table>                                                                                                                                  |                                                   |              |                      | 0010 | 01d0 | kkkk | kkkk |
| 0010              | 01d0                                                                                                                                                                                                                     | kkkk                                              | kkkk         |                      |      |      |      |      |
| Description:      | The contents of W are added to the contents of the register indicated by FSR2, offset by the value 'k'.<br>If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default). |                                                   |              |                      |      |      |      |      |
| Words:            | 1                                                                                                                                                                                                                        |                                                   |              |                      |      |      |      |      |
| Cycles:           | 1                                                                                                                                                                                                                        |                                                   |              |                      |      |      |      |      |
| Q Cycle Activity: |                                                                                                                                                                                                                          |                                                   |              |                      |      |      |      |      |
|                   | Q1                                                                                                                                                                                                                       | Q2                                                | Q3           | Q4                   |      |      |      |      |
|                   | Decode                                                                                                                                                                                                                   | Read 'k'                                          | Process Data | Write to destination |      |      |      |      |

**Example:** ADDWF [OFST], 0

Before Instruction

|                   |   |       |
|-------------------|---|-------|
| W                 | = | 17h   |
| OFST              | = | 2Ch   |
| FSR2              | = | 0A00h |
| Contents of 0A2Ch | = | 20h   |

After Instruction

|                   |   |     |
|-------------------|---|-----|
| W                 | = | 37h |
| Contents of 0A2Ch | = | 20h |

| BSF<br>(Indexed Literal Offset mode) |                                                                              |              |                      |      |      |
|--------------------------------------|------------------------------------------------------------------------------|--------------|----------------------|------|------|
| Syntax:                              | BSF [k], b                                                                   |              |                      |      |      |
| Operands:                            | $0 \leq f \leq 95$<br>$0 \leq b \leq 7$                                      |              |                      |      |      |
| Operation:                           | $1 \rightarrow ((FSR2) + k) \langle b \rangle$                               |              |                      |      |      |
| Status Affected:                     | None                                                                         |              |                      |      |      |
| Encoding:                            | <table><tr><td>1000</td><td>bbb0</td><td>kkkk</td><td>kkkk</td></tr></table> | 1000         | bbb0                 | kkkk | kkkk |
| 1000                                 | bbb0                                                                         | kkkk         | kkkk                 |      |      |
| Description:                         | Bit 'b' of the register indicated by FSR2, offset by the value 'k', is set.  |              |                      |      |      |
| Words:                               | 1                                                                            |              |                      |      |      |
| Cycles:                              | 1                                                                            |              |                      |      |      |
| Q Cycle Activity:                    |                                                                              |              |                      |      |      |
| Q1                                   | Q2                                                                           | Q3           | Q4                   |      |      |
| Decode                               | Read register 'f'                                                            | Process Data | Write to destination |      |      |

**Example:** BSF [FLAG\_OFST], 7

Before Instruction

|                   |   |       |
|-------------------|---|-------|
| FLAG_OFST         | = | 0Ah   |
| FSR2              | = | 0A00h |
| Contents of 0A0Ah | = | 55h   |

After Instruction

|                   |   |     |
|-------------------|---|-----|
| Contents of 0A0Ah | = | D5h |
|-------------------|---|-----|

## SETF Set Indexed (Indexed Literal Offset mode)

|                   |                                                                                                                                                           |              |                |      |      |        |          |              |                |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|----------------|------|------|--------|----------|--------------|----------------|
| Syntax:           | SETF [k]                                                                                                                                                  |              |                |      |      |        |          |              |                |
| Operands:         | $0 \leq k \leq 95$                                                                                                                                        |              |                |      |      |        |          |              |                |
| Operation:        | $FFh \rightarrow ((FSR2) + k)$                                                                                                                            |              |                |      |      |        |          |              |                |
| Status Affected:  | None                                                                                                                                                      |              |                |      |      |        |          |              |                |
| Encoding:         | <table><tr><td>0110</td><td>1000</td><td>kkkk</td><td>kkkk</td></tr></table>                                                                              | 0110         | 1000           | kkkk | kkkk |        |          |              |                |
| 0110              | 1000                                                                                                                                                      | kkkk         | kkkk           |      |      |        |          |              |                |
| Description:      | The contents of the register indicated by FSR2, offset by 'k', are set to FFh.                                                                            |              |                |      |      |        |          |              |                |
| Words:            | 1                                                                                                                                                         |              |                |      |      |        |          |              |                |
| Cycles:           | 1                                                                                                                                                         |              |                |      |      |        |          |              |                |
| Q Cycle Activity: |                                                                                                                                                           |              |                |      |      |        |          |              |                |
|                   | <table><tr><td>Q1</td><td>Q2</td><td>Q3</td><td>Q4</td></tr><tr><td>Decode</td><td>Read 'k'</td><td>Process Data</td><td>Write register</td></tr></table> | Q1           | Q2             | Q3   | Q4   | Decode | Read 'k' | Process Data | Write register |
| Q1                | Q2                                                                                                                                                        | Q3           | Q4             |      |      |        |          |              |                |
| Decode            | Read 'k'                                                                                                                                                  | Process Data | Write register |      |      |        |          |              |                |

**Example:** SETF [OFST]

Before Instruction

|                   |   |       |
|-------------------|---|-------|
| OFST              | = | 2Ch   |
| FSR2              | = | 0A00h |
| Contents of 0A2Ch | = | 00h   |

After Instruction

|                   |   |     |
|-------------------|---|-----|
| Contents of 0A2Ch | = | FFh |
|-------------------|---|-----|

## 42.2.5 SPECIAL CONSIDERATIONS WITH MICROCHIP MPLAB® IDE TOOLS

The latest versions of Microchip's software tools have been designed to fully support the extended instruction set of the PIC18(L)F25/26K83 family of devices. This includes the MPLAB C18 C compiler, MPASM assembly language and MPLAB Integrated Development Environment (IDE).

When selecting a target device for software development, MPLAB IDE will automatically set default Configuration bits for that device. The default setting for the XINST Configuration bit is '0', disabling the extended instruction set and Indexed Literal Offset Addressing mode. For proper execution of applications developed to take advantage of the extended instruction set, XINST must be set during programming.

To develop software for the extended instruction set, the user must enable support for the instructions and the Indexed Addressing mode in their language tool(s). Depending on the environment being used, this may be done in several ways:

- A menu option, or dialog box within the environment, that allows the user to configure the language tool and its settings for the project
- A command line option
- A directive in the source code

These options vary between different compilers, assemblers and development environments. Users are encouraged to review the documentation accompanying their development systems for the appropriate information.



## 43.0 REGISTER SUMMARY

**TABLE 43-1: REGISTER FILE SUMMARY FOR PIC18(L)F25/26K83 DEVICES**

| Addr          | Name     | Bit 7                                                                                                    | Bit 6           | Bit 5                                       | Bit 4                              | Bit 3  | Bit 2   | Bit 1   | Bit 0     | Register on page |     |
|---------------|----------|----------------------------------------------------------------------------------------------------------|-----------------|---------------------------------------------|------------------------------------|--------|---------|---------|-----------|------------------|-----|
| 3FFFh         | TOSU     | —                                                                                                        | —               | —                                           | Top-of-Stack Upper byte            |        |         |         |           | 28               |     |
| 3FFEh         | TOSH     | Top-of-Stack High byte                                                                                   |                 |                                             |                                    |        |         |         |           |                  | 28  |
| 3FFDh         | TOSL     | Top-of-Stack Low byte                                                                                    |                 |                                             |                                    |        |         |         |           |                  | 28  |
| 3FFCh         | STKPTR   | —                                                                                                        | —               | —                                           | Stack Pointer                      |        |         |         |           | 29               |     |
| 3FFBh         | PCLATU   | —                                                                                                        | —               | —                                           | Holding Register for PC Upper byte |        |         |         |           | 29               |     |
| 3FFAh         | PCLATH   | Holding Register for PC High byte                                                                        |                 |                                             |                                    |        |         |         |           |                  | 29  |
| 3FF9h         | PCL      | PC Low byte                                                                                              |                 |                                             |                                    |        |         |         |           |                  | 29  |
| 3FF8h         | TBLPTRU  | —                                                                                                        | —               | Program Memory Table Pointer Upper byte     |                                    |        |         |         |           | 29               |     |
| 3FF7h         | TBLPTRH  | Program Memory Table Pointer High byte                                                                   |                 |                                             |                                    |        |         |         |           |                  | 182 |
| 3FF6h         | TBLPTRL  | Program Memory Table Pointer Low byte                                                                    |                 |                                             |                                    |        |         |         |           |                  | 182 |
| 3FF5h         | TABLAT   | Table Latch                                                                                              |                 |                                             |                                    |        |         |         |           |                  | 182 |
| 3FF4h         | PRODH    | Product Register High byte                                                                               |                 |                                             |                                    |        |         |         |           |                  | 177 |
| 3FF3h         | PRODL    | Product Register Low byte                                                                                |                 |                                             |                                    |        |         |         |           |                  | 177 |
| 3FF2h         | —        | Unimplemented                                                                                            |                 |                                             |                                    |        |         |         |           |                  | —   |
| 3FF1h         | PCON1    | —                                                                                                        | —               | —                                           | —                                  | —      | —       | MEMV    | —         | 81               |     |
| 3FF0h         | PCON0    | STKOVF                                                                                                   | STKUNF          | WDTWV                                       | RWD $\overline{T}$                 | RMCLR  | RI      | POR     | BOR       | 80               |     |
| 3FEFh         | INDF0    | Uses contents of FSR0 to address data memory – value of FSR0 not changed                                 |                 |                                             |                                    |        |         |         |           | 50               |     |
| 3FEEh         | POSTINC0 | Uses contents of FSR0 to address data memory – value of FSR0 post-incremented                            |                 |                                             |                                    |        |         |         |           | 51               |     |
| 3FEDh         | POSTDEC0 | Uses contents of FSR0 to address data memory – value of FSR0 post-decremented                            |                 |                                             |                                    |        |         |         |           | 51               |     |
| 3FEC          | PREINC0  | Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented                             |                 |                                             |                                    |        |         |         |           | 51               |     |
| 3FEBh         | PLUSW0   | Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented – value of FSR0 offset by W |                 |                                             |                                    |        |         |         |           | 51               |     |
| 3FEAh         | FSR0H    | —                                                                                                        | —               | Indirect Data Memory Address Pointer 0 High |                                    |        |         |         |           | 51               |     |
| 3FE9h         | FSR0L    | Indirect Data Memory Address Pointer 0 Low                                                               |                 |                                             |                                    |        |         |         |           | 51               |     |
| 3FE8h         | WREG     | Working Register                                                                                         |                 |                                             |                                    |        |         |         |           | —                |     |
| 3FE7h         | INDF1    | Uses contents of FSR1 to address data memory – value of FSR1 not changed                                 |                 |                                             |                                    |        |         |         |           | 51               |     |
| 3FE6h         | POSTINC1 | Uses contents of FSR1 to address data memory – value of FSR1 post-incremented                            |                 |                                             |                                    |        |         |         |           | 51               |     |
| 3FE5h         | POSTDEC1 | Uses contents of FSR1 to address data memory – value of FSR1 post-decremented                            |                 |                                             |                                    |        |         |         |           | 51               |     |
| 3FE4h         | PREINC1  | Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented                             |                 |                                             |                                    |        |         |         |           | 51               |     |
| 3FE3h         | PLUSW1   | Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented – value of FSR1 offset by W |                 |                                             |                                    |        |         |         |           | 51               |     |
| 3FE2h         | FSR1H    | —                                                                                                        | —               | Indirect Data Memory Address Pointer 1 High |                                    |        |         |         |           | 51               |     |
| 3FE1h         | FSR1L    | Indirect Data Memory Address Pointer 1 Low                                                               |                 |                                             |                                    |        |         |         |           | 51               |     |
| 3FE0h         | BSR      | —                                                                                                        | —               | Bank Select Register                        |                                    |        |         |         |           | 34               |     |
| 3FDFh         | INDF2    | Uses contents of FSR2 to address data memory – value of FSR2 not changed                                 |                 |                                             |                                    |        |         |         |           | 51               |     |
| 3FDEh         | POSTINC2 | Uses contents of FSR2 to address data memory – value of FSR2 post-incremented                            |                 |                                             |                                    |        |         |         |           | 51               |     |
| 3FDDh         | POSTDEC2 | Uses contents of FSR2 to address data memory – value of FSR2 post-decremented                            |                 |                                             |                                    |        |         |         |           | 51               |     |
| 3FDC          | PREINC2  | Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented                             |                 |                                             |                                    |        |         |         |           | 51               |     |
| 3FDBh         | PLUSW2   | Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented – value of FSR2 offset by W |                 |                                             |                                    |        |         |         |           | 51               |     |
| 3FDAh         | FSR2H    | —                                                                                                        | —               | Indirect Data Memory Address Pointer 2 High |                                    |        |         |         |           | 51               |     |
| 3FD9h         | FSR2L    | Indirect Data Memory Address Pointer 2 Low                                                               |                 |                                             |                                    |        |         |         |           | 51               |     |
| 3FD8h         | STATUS   | —                                                                                                        | $\overline{TO}$ | $\overline{PD}$                             | N                                  | OV     | Z       | DC      | C         | 48               |     |
| 3FD7h         | IVTBASEU | —                                                                                                        | —               | —                                           | BASE20                             | BASE19 | BASE18  | BASE17  | BASE16    | 157              |     |
| 3FD6h         | IVTBASEH | BASE15                                                                                                   | BASE14          | BASE13                                      | BASE12                             | BASE11 | BASE10  | BASE9   | BASE8     | 157              |     |
| 3FD5h         | IVTBASEL | BASE7                                                                                                    | BASE6           | BASE5                                       | BASE4                              | BASE3  | BASE2   | BASE1   | BASE0     | 157              |     |
| 3FD4h         | IVTLOCK  | —                                                                                                        | —               | —                                           | —                                  | —      | —       | —       | IVTLOCKED | 159              |     |
| 3FD3h         | INTCON1  | STAT                                                                                                     |                 | —                                           | —                                  | —      | —       | —       | —         | 126              |     |
| 3FD2h         | INTCON0  | GIE                                                                                                      | GIEL            | IPEN                                        | —                                  | —      | INT2EDG | INT1EDG | INT0EDG   | 125              |     |
| 3FD1h - 3FD0h | —        | Unimplemented                                                                                            |                 |                                             |                                    |        |         |         |           | —                |     |

**Legend:** x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

**Note 1:** Not present in LF devices.

# PIC18(L)F25/26K83

**TABLE 43-1: REGISTER FILE SUMMARY FOR PIC18(L)F25/26K83 DEVICES (CONTINUED)**

| Addr          | Name   | Bit 7         | Bit 6  | Bit 5     | Bit 4  | Bit 3  | Bit 2     | Bit 1    | Bit 0  | Register on page |     |
|---------------|--------|---------------|--------|-----------|--------|--------|-----------|----------|--------|------------------|-----|
| 3FCEh         | PORTE  | —             | —      | —         | —      | RE3    | —         | —        | —      | 253              |     |
| 3FCDh         | —      | Unimplemented |        |           |        |        |           |          |        | —                |     |
| 3FCCh         | PORTC  | RC7           | RC6    | RC5       | RC4    | RC3    | RC2       | RC1      | RC0    | 253              |     |
| 3FCBh         | PORTB  | RB7           | RB6    | RB5       | RB4    | RB3    | RB2       | RB1      | RB0    | 253              |     |
| 3FCAh         | PORTA  | RA7           | RA6    | RA5       | RA4    | RA3    | RA2       | RA1      | RA0    | 253              |     |
| 3FC9h - 3FC5h | —      | Unimplemented |        |           |        |        |           |          |        | —                |     |
| 3FC4h         | TRISC  | TRISC7        | TRISC6 | TRISC5    | TRISC4 | TRISC3 | TRISC2    | TRISC1   | TRISC0 | 254              |     |
| 3FC3h         | TRISB  | TRISB7        | TRISB6 | TRISB5    | TRISB4 | TRISB3 | TRISB2    | TRISB1   | TRISB0 | 254              |     |
| 3FC2h         | TRISA  | TRISA7        | TRISA6 | TRISA5    | TRISA4 | TRISA3 | TRISA2    | TRISA1   | TRISA0 | 254              |     |
| 3FC1h - 3FBDh | —      | Unimplemented |        |           |        |        |           |          |        | —                |     |
| 3FBCh         | LATC   | LATC7         | LATC6  | LATC5     | LATC4  | LATC3  | LATC2     | LATC1    | LATC0  | 255              |     |
| 3FBBh         | LATB   | LATB7         | LATB6  | LATB5     | LATB4  | LATB3  | LATB2     | LATB1    | LATB0  | 255              |     |
| 3FBAh         | LATA   | LATA7         | LATA6  | LATA5     | LATA4  | LATA3  | LATA2     | LATA1    | LATA0  | 255              |     |
| 3FB9h         | T0CON1 | CS<2:0>       |        |           | ASYN   |        | CKPS<3:0> |          |        | 288              |     |
| 3FB8h         | T0CON0 | EN            | —      | OUT       | MD16   |        | OUTPS     |          |        | 287              |     |
| 3FB7h         | TMR0H  | TMR0H         |        |           |        |        |           |          |        | 289              |     |
| 3FB6h         | TMR0L  | TMR0L         |        |           |        |        |           |          |        | 289              |     |
| 3FB5h         | T1CLK  | CS            |        |           |        |        |           |          |        | 301              |     |
| 3FB4h         | T1GATE | GSS           |        |           |        |        |           |          |        | 302              |     |
| 3FB3h         | T1GCON | GE            | GPOL   | GTM       | GSPM   |        | GGO       | GVAL     | —      | —                | 300 |
| 3FB2h         | T1CON  | —             | —      | CKPS<1:0> |        |        | —         | SYNC     | RD16   | ON               | 324 |
| 3FB1h         | TMR1H  | TMR1H         |        |           |        |        |           |          |        | 303              |     |
| 3FB0h         | TMR1L  | TMR1L         |        |           |        |        |           |          |        | 303              |     |
| 3FAFh         | T2RST  | —             | —      | —         | RSEL   |        |           |          |        | 322              |     |
| 3FAEh         | T2CLK  | —             | —      | —         | —      | CS     |           |          |        | 301              |     |
| 3FADh         | T2HLT  | PSYNC         | CKPOL  | CKSYNC    | MODE   |        |           |          |        | 325              |     |
| 3FACH         | T2CON  | ON            | CKPS   |           |        | OUTPS  |           |          |        |                  | 299 |
| 3FABh         | T2PR   | PR2           |        |           |        |        |           |          |        | 323              |     |
| 3FAAh         | T2TMR  | TMR2          |        |           |        |        |           |          |        | 323              |     |
| 3FA9h         | T3CLK  | CS            |        |           |        |        |           |          |        | 301              |     |
| 3FA8h         | T3GATE | GSS           |        |           |        |        |           |          |        | 302              |     |
| 3FA7h         | T3GCON | GE            | GPOL   | GTM       | GSPM   |        | GGO       | GVAL     | —      | —                | 300 |
| 3FA6h         | T3CON  | —             | —      | CKPS      |        |        | —         | NOT_SYNC | RD16   | ON               | 324 |
| 3FA5h         | TMR3H  | TMR3H         |        |           |        |        |           |          |        | 303              |     |
| 3FA4h         | TMR3L  | TMR3L         |        |           |        |        |           |          |        | 303              |     |
| 3FA3h         | T4RST  | —             | —      | —         | RSEL   |        |           |          |        | 322              |     |
| 3FA2h         | T4CLK  | —             | —      | —         | —      | CS     |           |          |        | 321              |     |
| 3FA1h         | T4HLT  | PSYNC         | CKPOL  | CKSYNC    | MODE   |        |           |          |        | 325              |     |
| 3FA0h         | T4CON  | ON            | CKPS   |           |        | OUTPS  |           |          |        |                  | 324 |
| 3F9Fh         | T4PR   | PR4           |        |           |        |        |           |          |        | 323              |     |
| 3F9Eh         | T4TMR  | TMR4          |        |           |        |        |           |          |        | 323              |     |
| 3F9Dh         | T5CLK  | CS            |        |           |        |        |           |          |        | 321              |     |
| 3F9Ch         | T5GATE | GSS           |        |           |        |        |           |          |        | 302              |     |
| 3F9Bh         | T5GCON | GE            | GPOL   | GTM       | GSPM   |        | GGO       | GVAL     | —      | —                | 300 |
| 3F9Ah         | T5CON  | —             | —      | CKPS      |        |        | —         | NOT_SYNC | RD16   | ON               | 324 |
| 3F99h         | TMR5H  | TMR5H         |        |           |        |        |           |          |        | 303              |     |
| 3F98h         | TMR5L  | TMR5L         |        |           |        |        |           |          |        | 303              |     |
| 3F97h         | T6RST  | —             | —      | —         | RSEL   |        |           |          |        | 322              |     |

**Legend:** x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

**Note 1:** Not present in LF devices.

# PIC18(L)F25/26K83

**TABLE 43-1: REGISTER FILE SUMMARY FOR PIC18(L)F25/26K83 DEVICES (CONTINUED)**

| Addr  | Name     | Bit 7         | Bit 6    | Bit 5   | Bit 4    | Bit 3    | Bit 2    | Bit 1   | Bit 0   | Register on page |
|-------|----------|---------------|----------|---------|----------|----------|----------|---------|---------|------------------|
| 3F96h | T6CLK    | —             | —        | —       | —        | CS       |          |         |         | 301              |
| 3F95h | T6HLT    | PSYNC         | CKPOL    | CKSYNC  | MODE     |          |          |         |         | 325              |
| 3F94h | T6CON    | ON            | CKPS     |         |          | OUTPS    |          |         |         | 324              |
| 3F93h | T6PR     | PR6           |          |         |          |          |          |         |         | 323              |
| 3F92h | T6TMR    | TMR6          |          |         |          |          |          |         |         | 323              |
| 3F91h | ECANCON  | MDSEL1        | MDSEL0   | FIFOWM  | EWIN4    | EWIN3    | EWIN2    | EWIN1   | EWIN0   | 608              |
| 3F90h | COMSTAT  | RXB0OVFL      | RXB1OVFL | TXBO    | TXBP     | RXBP     | TXWARN   | RXWARN  | EWARN   | 609              |
| 3F90h | COMSTAT  | —             | RXBnOVFL | TXBO    | TXBP     | RXBP     | TXWARN   | RXWARN  | EWARN   | 609              |
| 3F90h | COMSTAT  | FIFOEMPTY     | RXBnOVFL | TXBO    | TXBP     | RXBP     | TXWARN   | RXWARN  | EWARN   | 609              |
| 3F8Fh | CANCON   | REQOP2        | REQOP1   | REQOP0  | ABAT     | WIN2     | WIN1     | WIN0    | —       | 604              |
| 3F8Fh | CANCON   | REQOP2        | REQOP1   | REQOP0  | ABAT     | —        | —        | —       | —       | 604              |
| 3F8Fh | CANCON   | REQOP2        | REQOP1   | REQOP0  | ABAT     | FP3      | FP2      | FP1     | FP0     | 604              |
| 3F8Eh | CANSTAT  | OPMODE2       | OPMODE1  | OPMODE0 | —        | ICODE2   | ICODE1   | ICODE0  | —       | 605              |
| 3F8Eh | CANSTAT  | OPMODE2       | OPMODE1  | OPMODE0 | EICODE4  | EICODE3  | EICODE2  | EICODE1 | EICODE0 | 605              |
| 3F8Dh | RXB0D7   | RXB0Dm7       | RXB0Dm6  | RXB0Dm5 | RXB0Dm4  | RXB0Dm3  | RXB0Dm2  | RXB0Dm1 | RXB0Dm0 | 621              |
| 3F8Ch | RXB0D6   | RXB0Dm7       | RXB0Dm6  | RXB0Dm5 | RXB0Dm4  | RXB0Dm3  | RXB0Dm2  | RXB0Dm1 | RXB0Dm0 | 621              |
| 3F8Bh | RXB0D5   | RXB0Dm7       | RXB0Dm6  | RXB0Dm5 | RXB0Dm4  | RXB0Dm3  | RXB0Dm2  | RXB0Dm1 | RXB0Dm0 | 621              |
| 3F8Ah | RXB0D4   | RXB0Dm7       | RXB0Dm6  | RXB0Dm5 | RXB0Dm4  | RXB0Dm3  | RXB0Dm2  | RXB0Dm1 | RXB0Dm0 | 621              |
| 3F89h | RXB0D3   | RXB0Dm7       | RXB0Dm6  | RXB0Dm5 | RXB0Dm4  | RXB0Dm3  | RXB0Dm2  | RXB0Dm1 | RXB0Dm0 | 621              |
| 3F88h | RXB0D2   | RXB0Dm7       | RXB0Dm6  | RXB0Dm5 | RXB0Dm4  | RXB0Dm3  | RXB0Dm2  | RXB0Dm1 | RXB0Dm0 | 621              |
| 3F87h | RXB0D1   | RXB0Dm7       | RXB0Dm6  | RXB0Dm5 | RXB0Dm4  | RXB0Dm3  | RXB0Dm2  | RXB0Dm1 | RXB0Dm0 | 621              |
| 3F86h | RXB0D0   | RXB0Dm7       | RXB0Dm6  | RXB0Dm5 | RXB0Dm4  | RXB0Dm3  | RXB0Dm2  | RXB0Dm1 | RXB0Dm0 | 621              |
| 3F85h | RXB0DLC  | —             | RXRTR    | RB1     | RB0      | DLC3     | DLC2     | DLC1    | DLC0    | 621              |
| 3F84h | RXB0EIDL | EID7          | EID6     | EID5    | EID4     | EID3     | EID2     | EID1    | EID0    | 621              |
| 3F83h | RXB0EIDH | EID15         | EID14    | EID13   | EID12    | EID11    | EID10    | EID9    | EID8    | 621              |
| 3F82h | RXB0SIDL | SID2          | SID1     | SID0    | SRR      | EXID     | —        | EID17   | EID16   | 621              |
| 3F81h | RXB0SIDH | SID10         | SID9     | SID8    | SID7     | SID6     | SID5     | SID4    | SID3    | 621              |
| 3F80h | RXB0CON  | RXFUL         | RXM1     | RXM0    | —        | RXRTRRO  | RXB0DBEN | JTOFF   | FILHIT0 | 621              |
| 3F80h | RXB0CON  | RXFUL         | RXM1     | RTRRO   | FILHITF4 | FILHIT3  | FILHIT2  | FILHIT1 | FILHIT0 | 621              |
| 3F7Fh | CCP1CAP  | —             | —        | —       | —        | CTS<3:0> |          |         |         | 339              |
| 3F7Eh | CCP1CON  | EN            | —        | OUT     | FMT      | MODE     |          |         |         | 336              |
| 3F7Dh | CCPR1H   | RH            |          |         |          |          |          |         |         | 340              |
| 3F7Ch | CCPR1L   | RL            |          |         |          |          |          |         |         | 339              |
| 3F7Bh | CCP2CAP  | —             | —        | —       | —        | CTS<3:0> |          |         |         | 339              |
| 3F7Ah | CCP2CON  | EN            | —        | OUT     | FMT      | MODE     |          |         |         | 336              |
| 3F79h | CCPR2H   | RH            |          |         |          |          |          |         |         | 340              |
| 3F78h | CCPR2L   | RL            |          |         |          |          |          |         |         | 339              |
| 3F77h | CCP3CAP  | —             | —        | —       | —        | CTS<3:0> |          |         |         | 339              |
| 3F76h | CCP3CON  | EN            | —        | OUT     | FMT      | MODE     |          |         |         | 336              |
| 3F75h | CCPR3H   | RH            |          |         |          |          |          |         |         | 340              |
| 3F74h | CCPR3L   | RL            |          |         |          |          |          |         |         | 339              |
| 3F73h | CCP4CAP  | —             | —        | —       | —        | CTS<3:0> |          |         |         | 339              |
| 3F72h | CCP4CON  | EN            | —        | OUT     | FMT      | MODE     |          |         |         | 336              |
| 3F71h | CCPR4H   | RH            |          |         |          |          |          |         |         | 340              |
| 3F70h | CCPR4L   | RL            |          |         |          |          |          |         |         | 339              |
| 3F6Fh | —        | Unimplemented |          |         |          |          |          |         |         | —                |
| 3F6Eh | PWM5CON  | EN            | —        | OUT     | POL      | —        | —        | —       | —       | 345              |
| 3F6Dh | PWM5DCH  | DC9           | DC8      | DC7     | DC6      | DC5      | DC4      | DC3     | DC2     | 347              |
| 3F6Ch | PWM5DCL  | DC1           | DC0      | —       | —        | —        | —        | —       | —       | 347              |
| 3F6Bh | —        | Unimplemented |          |         |          |          |          |         |         | —                |

**Legend:** x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

**Note 1:** Not present in LF devices.

# PIC18(L)F25/26K83

**TABLE 43-1: REGISTER FILE SUMMARY FOR PIC18(L)F25/26K83 DEVICES (CONTINUED)**

| Addr          | Name      | Bit 7         | Bit 6 | Bit 5  | Bit 4 | Bit 3  | Bit 2 | Bit 1  | Bit 0 | Register on page |
|---------------|-----------|---------------|-------|--------|-------|--------|-------|--------|-------|------------------|
| 3F6Ah         | PWM6CON   | EN            | —     | OUT    | POL   | —      | —     | —      | —     | 345              |
| 3F69h         | PWM6DCH   | DC9           |       | DC7    | DC6   | DC5    | DC4   | DC3    | DC2   | 347              |
| 3F68h         | PWM6DCL   | DC1           | DC0   | —      | —     | —      | —     | —      | —     | 347              |
| 3F67h         | —         | Unimplemented |       |        |       |        |       |        |       | —                |
| 3F66h         | PWM7CON   | EN            | —     | OUT    | POL   | —      | —     | —      | —     | 345              |
| 3F65h         | PWM7DCH   | DC9           | DC8   | DC7    | DC6   | DC5    | DC4   | DC3    | DC2   | 347              |
| 3F64h         | PWM7DCL   | DC1           | DC0   | —      | —     | —      | —     | —      | —     | 347              |
| 3F63h         | —         | Unimplemented |       |        |       |        |       |        |       | —                |
| 3F62h         | PWM8CON   | EN            | —     | OUT    | POL   | —      | —     | —      | —     | 345              |
| 3F61h         | PWM8DCH   | DC9           | DC8   | DC7    | DC6   | DC5    | DC4   | DC3    | DC2   | 347              |
| 3F60h         | PWM8DCL   | DC1           | DC0   | —      | —     | —      | —     | —      | —     | 347              |
| 3F5Fh         | CCPTMRS1  | P8TSEL        |       | P7TSEL |       | P6TSEL |       | P5TSEL |       | 346              |
| 3F5Eh         | CCPTMRS0  | C4TSEL        |       | C3TSEL |       | C2TSEL |       | C1TSEL |       | 346              |
| 3F5Dh - 3F5Bh | —         | Unimplemented |       |        |       |        |       |        |       | —                |
| 3F5Ah         | CWG1STR   | OVRD          | OVRC  | OVRB   | OVRA  | STRD   | STRC  | STRB   | STRA  | 415              |
| 3F59h         | CWG1AS1   | —             | AS6E  | AS5E   | AS4E  | AS3E   | AS2E  | AS1E   | AS0E  | 417              |
| 3F58h         | CWG1AS0   | SHUTDOWN      | REN   | LSBD   |       | LSAC   |       | —      | —     | 416              |
| 3F57h         | CWG1CON1  | —             | —     | IN     | —     | POLD   | POLC  | POLB   | POLA  | 412              |
| 3F56h         | CWG1CON0  | EN            | LD    | —      | —     | —      | MODE  |        |       | 411              |
| 3F55h         | CWG1DBF   | —             | —     | DBF    |       |        |       |        |       | 418              |
| 3F54h         | CWG1DBR   | —             | —     | DBR    |       |        |       |        |       | 418              |
| 3F53h         | CWG1ISM   | —             | —     | —      | —     | IS     |       |        |       | 414              |
| 3F52h         | CWG1CLK   | —             | —     | —      | —     | —      | —     | —      | CS    | 413              |
| 3F51h         | CWG2STR   | OVRD          | OVRC  | OVRB   | OVRA  | STRD   | STRC  | STRB   | STRA  | 415              |
| 3F50h         | CWG2AS1   | —             | AS6E  | AS5E   | AS4E  | AS3E   | AS2E  | AS1E   | AS0E  | 417              |
| 3F4Fh         | CWG2AS0   | SHUTDOWN      | REN   | LSBD   |       | LSAC   |       | —      | —     | 416              |
| 3F4Eh         | CWG2CON1  | —             | —     | IN     | —     | POLD   | POLC  | POLB   | POLA  | 412              |
| 3F4Dh         | CWG2CON0  | EN            | LD    | —      | —     | —      | MODE  |        |       | 411              |
| 3F4Ch         | CWG2DBF   | —             | —     | DBF    |       |        |       |        |       | 418              |
| 3F4Bh         | CWG2DBR   | —             | —     | DBR    |       |        |       |        |       | 418              |
| 3F4Ah         | CWG2ISM   | —             | —     | —      | —     | IS     |       |        |       | 414              |
| 3F49h         | CWG2CLK   | —             | —     | —      | —     | —      | —     | —      | CS    | 413              |
| 3F48h         | CWG3STR   | OVRD          | OVRC  | OVRB   | OVRA  | STRD   | STRC  | STRB   | STRA  | 415              |
| 3F47h         | CWG3AS1   | —             | AS6E  | AS5E   | AS4E  | AS3E   | AS2E  | AS1E   | AS0E  | 417              |
| 3F46h         | CWG3AS0   | SHUTDOWN      | REN   | LSBD   |       | LSAC   |       | —      | —     | 416              |
| 3F45h         | CWG3CON1  | —             | —     | IN     | —     | POLD   | POLC  | POLB   | POLA  | 412              |
| 3F44h         | CWG3CON0  | EN            | LD    | —      | —     | —      | MODE  |        |       | 411              |
| 3F43h         | CWG3DBF   | —             | —     | DBF    |       |        |       |        |       | 418              |
| 3F42h         | CWG3DBR   | —             | —     | DBR    |       |        |       |        |       | 418              |
| 3F41h         | CWG3ISM   | —             | —     | —      | —     | IS     |       |        |       | 414              |
| 3F40h         | CWG3CLK   | —             | —     | —      | —     | —      | —     | —      | CS    | 413              |
| 3F3Fh         | NCO1CLK   | PWS           |       |        | —     | CKS    |       |        |       | 441              |
| 3F3Eh         | NCO1CON   | EN            | —     | OUT    | POL   | —      | —     | —      | PFM   | 440              |
| 3F3Dh         | NCO1INCUI | INC           |       |        |       |        |       |        |       | 444              |
| 3F3Ch         | NCO1INCH  | INC           |       |        |       |        |       |        |       | 443              |
| 3F3Bh         | NCO1INCL  | INC           |       |        |       |        |       |        |       | 443              |
| 3F3Ah         | NCO1ACCU  | ACC           |       |        |       |        |       |        |       | 443              |
| 3F39h         | NCO1ACCH  | ACC           |       |        |       |        |       |        |       | 442              |
| 3F38h         | NCO1ACCL  | ACC           |       |        |       |        |       |        |       | 442              |

**Legend:** x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

**Note 1:** Not present in LF devices.

# PIC18(L)F25/26K83

**TABLE 43-1: REGISTER FILE SUMMARY FOR PIC18(L)F25/26K83 DEVICES (CONTINUED)**

| Addr          | Name     | Bit 7         | Bit 6  | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |     |
|---------------|----------|---------------|--------|-------|-------|-------|-------|-------|-------|------------------|-----|
| 3F37h - 3F24h | —        | Unimplemented |        |       |       |       |       |       |       | —                |     |
| 3F23h         | SMT1WIN  | —             | —      | —     | WSEL  |       |       |       |       | 385              |     |
| 3F22h         | SMT1SIG  | —             | —      | —     | SSEL  |       |       |       |       | 386              |     |
| 3F21h         | SMT1CLK  | —             | —      | —     | —     | —     | CSEL  |       |       | 384              |     |
| 3F20h         | SMT1STAT | CPRUP         | CPWUP  | RST   | —     | —     | TS    | WS    | AS    | 383              |     |
| 3F1Fh         | SMT1CON1 | GO            | REPEAT | —     | —     | MODE  |       |       |       | 382              |     |
| 3F1Eh         | SMT1CON0 | EN            | —      | STP   | WPOL  | SPOL  | CPOL  | PS    |       | 381              |     |
| 3F1Dh         | SMT1PRU  | PR            |        |       |       |       |       |       |       | 388              |     |
| 3F1Ch         | SMT1PRH  | PR            |        |       |       |       |       |       |       | 390              |     |
| 3F1Bh         | SMT1PRL  | PR            |        |       |       |       |       |       |       | 390              |     |
| 3F1Ah         | SMT1CPWU | CPW           |        |       |       |       |       |       |       | 389              |     |
| 3F19h         | SMT1CPWH | CPW           |        |       |       |       |       |       |       | 389              |     |
| 3F18h         | SMT1CPWL | CPW           |        |       |       |       |       |       |       | 389              |     |
| 3F17h         | SMT1CPRU | CPR           |        |       |       |       |       |       |       | 388              |     |
| 3F16h         | SMT1CPRH | CPR           |        |       |       |       |       |       |       | 388              |     |
| 3F15h         | SMT1CPRL | CPR           |        |       |       |       |       |       |       | 388              |     |
| 3F14h         | SMT1TMRU | TMR           |        |       |       |       |       |       |       | 387              |     |
| 3F13h         | SMT1TMRH | TMR           |        |       |       |       |       |       |       | 387              |     |
| 3F12h         | SMT1TMRL | TMR           |        |       |       |       |       |       |       | 387              |     |
| 3F11h         | SMT2WIN  | —             | —      | —     | WSEL  |       |       |       |       | 385              |     |
| 3F10h         | SMT2SIG  | —             | —      | —     | SSEL  |       |       |       |       | 386              |     |
| 3F0Fh         | SMT2CLK  | —             | —      | —     | —     | —     | CSEL  |       |       | 384              |     |
| 3F0Eh         | SMT2STAT | CPRUP         | CPWUP  | RST   | —     | —     | TS    | WS    | AS    | 383              |     |
| 3F0Dh         | SMT2CON1 | GO            | REPEAT | —     | —     | MODE  |       |       |       | 382              |     |
| 3F0Ch         | SMT2CON0 | EN            | —      | STP   | WPOL  | SPOL  | CPOL  | PS    |       | 381              |     |
| 3F0Bh         | SMT2PRU  | PR            |        |       |       |       |       |       |       | 388              |     |
| 3F0Ah         | SMT2PRH  | PR            |        |       |       |       |       |       |       | 390              |     |
| 3F09h         | SMT2PRL  | PR            |        |       |       |       |       |       |       | 390              |     |
| 3F08h         | SMT2CPWU | CPW           |        |       |       |       |       |       |       | 389              |     |
| 3F07h         | SMT2CPWH | CPW           |        |       |       |       |       |       |       | 389              |     |
| 3F06h         | SMT2CPWL | CPW           |        |       |       |       |       |       |       | 389              |     |
| 3F05h         | SMT2CPRU | CPR           |        |       |       |       |       |       |       | 388              |     |
| 3F04h         | SMT2CPRH | CPR           |        |       |       |       |       |       |       | 388              |     |
| 3F03h         | SMT2CPRL | CPR           |        |       |       |       |       |       |       | 388              |     |
| 3F02h         | SMT2TMRU | TMR           |        |       |       |       |       |       |       | 387              |     |
| 3F01h         | SMT2TMRH | TMR           |        |       |       |       |       |       |       | 387              |     |
| 3F00h         | SMT2TMRL | TMR           |        |       |       |       |       |       |       | 387              |     |
| 3EFFh         | ADCLK    | —             | —      | CS    |       |       |       |       |       | 676              |     |
| 3EFEh         | ADACT    | —             | —      | —     | ACT   |       |       |       |       | 676              |     |
| 3EFDh         | ADREF    | NREF          |        |       |       | PREF  |       |       |       | 676              |     |
| 3EFCh         | ADSTAT   | ADAOV         | UTHR   | LTHR  | MATH  | —     | STAT  |       |       | 675              |     |
| 3EFBh         | ADCON3   | —             | CALC   |       |       |       | SOI   | TMD   |       |                  | 674 |
| 3EFAh         | ADCON2   | PSIS          | CRS    |       |       |       | ACLRL | MODE  |       |                  | 673 |
| 3EF9h         | ADCON1   | PPOL          | IPEN   | GPOL  | —     | —     | —     | —     | DSEN  | 672              |     |
| 3EF8h         | ADCON0   | ON            | CONT   | —     | CS    | FM    |       | —     | GO    | 671              |     |
| 3EF7h         | ADPREH   | —             | —      | —     | PRE   |       |       |       |       | 678              |     |
| 3EF6h         | ADPREL   | PRE           |        |       |       |       |       |       |       | 678              |     |
| 3EF5h         | ADCAP    | —             | —      | —     | ADCAP |       |       |       |       | 680              |     |
| 3EF4h         | ADACQH   | —             | —      | —     | ACQ   |       |       |       |       | 679              |     |

**Legend:** x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

**Note 1:** Not present in LF devices.

# PIC18(L)F25/26K83

**TABLE 43-1: REGISTER FILE SUMMARY FOR PIC18(L)F25/26K83 DEVICES (CONTINUED)**

| Addr          | Name     | Bit 7         | Bit 6 | Bit 5  | Bit 4 | Bit 3  | Bit 2  | Bit 1  | Bit 0 | Register on page |
|---------------|----------|---------------|-------|--------|-------|--------|--------|--------|-------|------------------|
| 3EF3h         | ADACQL   | ACQ           |       |        |       |        |        |        |       | 679              |
| 3EF2h         | —        | Unimplemented |       |        |       |        |        |        |       | —                |
| 3EF1h         | ADPCH    | —             | —     | ADPCH  |       |        |        |        |       | 677              |
| 3EF0h         | ADRESH   | RES           |       |        |       |        |        |        |       | 682              |
| 3EEFh         | ADRESL   | RES           |       |        |       |        |        |        |       | 682              |
| 3EEEh         | ADPREVH  | PREV          |       |        |       |        |        |        |       | 684              |
| 3EEDh         | ADPREVL  | PREV          |       |        |       |        |        |        |       | 684              |
| 3EECh         | ADRPT    | RPT           |       |        |       |        |        |        |       | 680              |
| 3EEBh         | ADCNT    | CNT           |       |        |       |        |        |        |       | 681              |
| 3EEAh         | ADACCU   | ACC           |       |        |       |        |        |        |       | 685              |
| 3EE9h         | ADACCH   | ACC           |       |        |       |        |        |        |       | 685              |
| 3EE8h         | ADACCL   | ACC           |       |        |       |        |        |        |       | 685              |
| 3EE7h         | ADFLTRH  | FLTR          |       |        |       |        |        |        |       | 681              |
| 3EE6h         | ADFLTRL  | FLTR          |       |        |       |        |        |        |       | 681              |
| 3EE5h         | ADSTPTH  | STPT          |       |        |       |        |        |        |       | 686              |
| 3EE4h         | ADSTPTL  | STPT          |       |        |       |        |        |        |       | 686              |
| 3EE3h         | ADERRH   | ERR           |       |        |       |        |        |        |       | 687              |
| 3EE2h         | ADERRL   | ERR           |       |        |       |        |        |        |       | 687              |
| 3EE1h         | ADUTHH   | UTH           |       |        |       |        |        |        |       | 688              |
| 3EE0h         | ADUTHL   | UTH           |       |        |       |        |        |        |       | 688              |
| 3EDFh         | ADLTHH   | LTH           |       |        |       |        |        |        |       | 687              |
| 3EDEh         | ADLTHL   | LTH           |       |        |       |        |        |        |       | 688              |
| 3EDDh - 3ED8h | —        | Unimplemented |       |        |       |        |        |        |       | —                |
| 3ED7h         | ADCP     | ON            | —     | —      | —     | —      | —      | —      | CPRDY | 690              |
| 3ED6h - 3ECBh | —        | Unimplemented |       |        |       |        |        |        |       | —                |
| 3ECAh         | HLVDCON1 | —             | —     | —      | —     | SEL    |        |        |       | 712              |
| 3EC9h         | HLVDCON0 | EN            | —     | OUT    | RDY   | —      | —      | INTH   | INTL  | 711              |
| 3EC8h - 3EC4h | —        | Unimplemented |       |        |       |        |        |        |       | —                |
| 3EC3h         | ZCDCON   | SEN           | —     | OUT    | POL   | —      | —      | INTP   | INTN  | 449              |
| 3EC2h         | —        | Unimplemented |       |        |       |        |        |        |       | —                |
| 3EC1h         | FVRCON   | EN            | RDY   | TSEN   | TSRNG | CDAFVR |        | ADFVR  |       | 651              |
| 3EC0h         | CMOUT    | —             | —     | —      | —     | —      | —      | C2OUT  | C1OUT | 704              |
| 3EBFh         | CM1PCH   | —             | —     | —      | —     | —      | PCH    |        |       | 704              |
| 3EBEh         | CM1NCH   | —             | —     | —      | —     | —      | NCH    |        |       | 703              |
| 3EBDh         | CM1CON1  | —             | —     | —      | —     | —      | —      | INTP   | INTN  | 703              |
| 3EBCh         | CM1CON0  | EN            | OUT   | —      | POL   | —      | —      | HYS    | SYNC  | 702              |
| 3EBBh         | CM2PCH   | —             | —     | —      | —     | —      | PCH    |        |       | 704              |
| 3EBAh         | CM2NCH   | —             | —     | —      | —     | —      | NCH    |        |       | 703              |
| 3EB9h         | CM2CON1  | —             | —     | —      | —     | —      | —      | INTP   | INTN  | 703              |
| 3EB8h         | CM2CON0  | EN            | OUT   | —      | POL   | —      | —      | HYS    | SYNC  | 702              |
| 3EB7h - 3E9Fh | —        | Unimplemented |       |        |       |        |        |        |       | —                |
| 3E9Eh         | DAC1CON0 | EN            | —     | OE1    | OE2   | PSS    |        | —      | NSS   | 694              |
| 3E9Dh         | —        | Unimplemented |       |        |       |        |        |        |       | —                |
| 3E9Ch         | DAC1CON1 | —             | —     | —      | DATA  |        |        |        |       | 695              |
| 3E9Bh - 3DFBh | —        | Unimplemented |       |        |       |        |        |        |       |                  |
| 3DFAh         | U1ERRIE  | TXMTIE        | PERIE | ABDOVE | CERIE | FERIE  | RXBKIE | RXFOIE | TXCIE | 488              |

**Legend:** x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

**Note 1:** Not present in LF devices.

# PIC18(L)F25/26K83

**TABLE 43-1: REGISTER FILE SUMMARY FOR PIC18(L)F25/26K83 DEVICES (CONTINUED)**

| Addr          | Name      | Bit 7         | Bit 6  | Bit 5   | Bit 4  | Bit 3  | Bit 2  | Bit 1  | Bit 0  | Register on page |
|---------------|-----------|---------------|--------|---------|--------|--------|--------|--------|--------|------------------|
| 3DF9h         | U1ERRIR   | TXMTIF        | PERIF  | ABDOVF  | CERIF  | FERIF  | RXBKIF | RXFOIF | TXCIF  | 487              |
| 3DF8h         | U1UIR     | WUIF          | ABDIF  | —       | —      | —      | ABDIE  | —      | —      | 489              |
| 3DF7h         | U1FIFO    | TXWRE         | STPMD  | TXBE    | TXBF   | RXIDL  | XON    | RXBE   | RXBF   | 490              |
| 3DF6h         | U1BRGH    | BRGH          |        |         |        |        |        |        |        | 491              |
| 3DF5h         | U1BRGL    | BRGL          |        |         |        |        |        |        |        | 491              |
| 3DF4h         | U1CON2    | RUNOVF        | RXPOL  | STP     |        | C0EN   | TXPOL  | FLO    |        | 486              |
| 3DF3h         | U1CON1    | ON            | —      | —       | WUE    | RXBIMD | —      | BRKOVF | SENDB  | 485              |
| 3DF2h         | U1CON0    | BRGS          | ABDEN  | TXEN    | RXEN   | MODE   |        |        |        | 484              |
| 3DF1h         | U1P3H     | —             | —      | —       | —      | —      | —      | —      | P3H    | 495              |
| 3DF0h         | U1P3L     | P3L           |        |         |        |        |        |        |        | 495              |
| 3DEFh         | U1P2H     | —             | —      | —       | —      | —      | —      | —      | P2H    | 494              |
| 3DEEh         | U1P2L     | P2L           |        |         |        |        |        |        |        | 494              |
| 3DEDh         | U1P1H     | —             | —      | —       | —      | —      | —      | —      | P1H    | 493              |
| 3DECh         | U1P1L     | P1L           |        |         |        |        |        |        |        | 493              |
| 3DEBh         | U1TXCHK   | TXCHK         |        |         |        |        |        |        |        | 496              |
| 3DEAh         | U1TXB     | TXB           |        |         |        |        |        |        |        | 492              |
| 3DE9h         | U1RXCHK   | RXCHK         |        |         |        |        |        |        |        | 496              |
| 3DE8h         | U1RXB     | RXB           |        |         |        |        |        |        |        | 492              |
| 3DE7h - 3DE3h | —         | Unimplemented |        |         |        |        |        |        |        | —                |
| 3DE2h         | U2ERRIE   | TXMTIE        | PERIE  | ABDOVF  | CERIE  | FERIE  | RXBKIE | RXFOIE | TXCIE  | 488              |
| 3DE1h         | U2ERRIR   | TXMTIF        | PERIF  | ABDOVF  | CERIF  | FERIF  | RXBKIF | RXFOIF | TXCIF  | 487              |
| 3DE0h         | U2UIR     | WUIF          | ABDIF  | —       | —      | —      | ABDIE  | —      | —      | 489              |
| 3DDFh         | U2FIFO    | TXWRE         | STPMD  | TXBE    | TXBF   | RXIDL  | XON    | RXBE   | RXBF   | 490              |
| 3DDEh         | U2BRGH    | BRGH          |        |         |        |        |        |        |        | 491              |
| 3DDCh         | U2BRGL    | BRGL          |        |         |        |        |        |        |        | 491              |
| 3DDh          | U2CON2    | RUNOVF        | RXPOL  | STP     |        | —      | TXPOL  | FLO    |        | 486              |
| 3DDh          | U2CON1    | ON            | —      | —       | WUE    | RXBIMD | —      | BRKOVF | SENDB  | 485              |
| 3DDh          | U2CON0    | BRGS          | ABDEN  | TXEN    | RXEN   | MODE   |        |        |        | 484              |
| 3DD9h         | U2P3H     | —             | —      | —       | —      | —      | —      | —      | P3H    | 495              |
| 3DD8h         | U2P3L     | P3L           |        |         |        |        |        |        |        | 495              |
| 3DD7h         | U2P2H     | —             | —      | —       | —      | —      | —      | —      | P2H    | 494              |
| 3DD6h         | U2P2L     | P2L           |        |         |        |        |        |        |        | 494              |
| 3DD5h         | U2P1H     | —             | —      | —       | —      | —      | —      | —      | P1H    | 493              |
| 3DD4h         | U2P1L     | P1L           |        |         |        |        |        |        |        | 493              |
| 3DD3h         | U2TXCHK   | TXCHK         |        |         |        |        |        |        |        | 496              |
| 3DD2h         | U2TXB     | TXB           |        |         |        |        |        |        |        | 492              |
| 3DD1h         | U2RXCHK   | RXCHK         |        |         |        |        |        |        |        | 496              |
| 3DD0h         | U2RXB     | RXB           |        |         |        |        |        |        |        | 492              |
| 3DCFh - 3D7Dh | —         | Unimplemented |        |         |        |        |        |        |        | —                |
| 3D7Ch         | I2C1BTO   | BTO           |        |         |        |        |        |        |        | 569              |
| 3D7Bh         | I2C1CLK   | CLK           |        |         |        |        |        |        |        | 568              |
| 3D7Ah         | I2C1PIE   | CNTIE         | ACKTIE | —       | WRIE   | ADRIE  | PCIE   | RSCIE  | SCIE   | 575              |
| 3D79h         | I2C1PIR   | CNTIF         | ACKTIF | —       | WRIF   | ADRIF  | PCIF   | RSCIF  | SCIF   | 574              |
| 3D78h         | I2C1STAT1 | TXWE          | —      | TXBE    | —      | RXRE   | CLRBF  | —      | RXBF   | 571              |
| 3D77h         | I2C1STAT0 | BFRE          | SMA    | MMA     | R      | D      | —      | —      | —      | 570              |
| 3D76h         | I2C1ERR   | —             | BTOIF  | BCLIF   | NACKIF | —      | BTOIE  | BCLIE  | NACKIE | 572              |
| 3D75h         | I2C1CON2  | ACNT          | GCEN   | FME     | ABD    | SDAHT  |        | BFRET  |        | 567              |
| 3D74h         | I2C1CON1  | ACKCNT        | ACKDT  | ACKSTAT | ACKT   | —      | RXO    | TXU    | CSD    | 566              |

**Legend:** x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

**Note 1:** Not present in LF devices.

# PIC18(L)F25/26K83

**TABLE 43-1: REGISTER FILE SUMMARY FOR PIC18(L)F25/26K83 DEVICES (CONTINUED)**

| Addr          | Name       | Bit 7         | Bit 6  | Bit 5   | Bit 4  | Bit 3 | Bit 2  | Bit 1 | Bit 0  | Register on page |
|---------------|------------|---------------|--------|---------|--------|-------|--------|-------|--------|------------------|
| 3D73h         | I2C1CON0   | EN            | RSEN   | S       | CSTR   | MDR   | MODE   |       |        | 564              |
| 3D72h         | I2C1ADR3   | ADR           |        |         |        |       |        |       | —      | 579              |
| 3D71h         | I2C1ADR2   | ADR           |        |         |        |       |        |       | —      | 578              |
| 3D70h         | I2C1ADR1   | ADR           |        |         |        |       |        |       | —      | 577              |
| 3D6Fh         | I2C1ADR0   | ADR           |        |         |        |       |        |       | —      | 576              |
| 3D6Eh         | I2C1ADB1   | ADB           |        |         |        |       |        |       | —      | 581              |
| 3D6Dh         | I2C1ADB0   | ADB           |        |         |        |       |        |       | —      | 580              |
| 3D6Ch         | I2C1CNT    | CNT           |        |         |        |       |        |       | —      | 573              |
| 3D6Bh         | I2C1TXB    | TXB           |        |         |        |       |        |       | —      | —                |
| 3D6Ah         | I2C1RXB    | RXB           |        |         |        |       |        |       | —      | —                |
| 3D69h - 3D67h | —          | Unimplemented |        |         |        |       |        |       | —      | —                |
| 3D66h         | I2C2BTO    | BTO           |        |         |        |       |        |       | —      | 569              |
| 3D65h         | I2C2CLK    | CLK           |        |         |        |       |        |       | —      | 568              |
| 3D64h         | I2C2PIE    | CNTIE         | ACKTIE | —       | WRIE   | ADRIE | PCIE   | RSCIE | SCIE   | 575              |
| 3D63h         | I2C2PIR    | CNTIF         | ACKTIF | —       | WRIF   | ADRIF | PCIF   | RSCIF | SCIF   | 574              |
| 3D62h         | I2C2STAT1  | TXWE          | —      | TXBE    | —      | RXRE  | CLRBFB | —     | RXBF   | 571              |
| 3D61h         | I2C2STAT0  | BFRE          | SMA    | MMA     | R      | D     | —      | —     | —      | 570              |
| 3D60h         | I2C2ERR    | —             | BTOIF  | BCLIF   | NACKIF | —     | BTOIE  | BCLIE | NACKIE | 572              |
| 3D5Fh         | I2C2CON2   | ACNT          | GCEN   | FME     | ABD    | SDAHT |        | BFRET |        | 567              |
| 3D5Eh         | I2C2CON1   | ACKCNT        | ACKDT  | ACKSTAT | ACKT   | —     | RXO    | TXU   | CSD    | 566              |
| 3D5Dh         | I2C2CON0   | EN            | RSEN   | S       | CSTR   | MDR   | MODE   |       |        | 564              |
| 3D5Ch         | I2C2ADR3   | ADR           |        |         |        |       |        |       | —      | 579              |
| 3D5Bh         | I2C2ADR2   | ADR           |        |         |        |       |        |       | —      | 578              |
| 3D5Ah         | I2C2ADR1   | ADR           |        |         |        |       |        |       | —      | 577              |
| 3D59h         | I2C2ADR0   | ADR           |        |         |        |       |        |       | —      | 576              |
| 3D58h         | I2C2ADB1   | ADB           |        |         |        |       |        |       | —      | 581              |
| 3D57h         | I2C2ADB0   | ADB           |        |         |        |       |        |       | —      | 580              |
| 3D56h         | I2C2CNT    | CNT           |        |         |        |       |        |       | —      | 573              |
| 3D55h         | I2C2TXB    | TXB           |        |         |        |       |        |       | —      | —                |
| 3D54h         | I2C2RXB    | RXB           |        |         |        |       |        |       | —      | —                |
| 3D53h - 3D1Dh | —          | Unimplemented |        |         |        |       |        |       | —      | —                |
| 3D1Ch         | SPI1CLK    | CLKSEL        |        |         |        |       |        |       | —      | 528              |
| 3D1Bh         | SPI1INTE   | SRMTIE        | TCZIE  | SOSIE   | EOSIE  | —     | RXOIE  | TXUIE | —      | 522              |
| 3D1Ah         | SPI1INTF   | SRMTIF        | TCZIF  | SOSIF   | EOSIF  | —     | RXOIF  | TXUIF | —      | 521              |
| 3D19h         | SPI1BAUD   | BAUD          |        |         |        |       |        |       | —      | 524              |
| 3D18h         | SPI1TWIDTH | —             | —      | —       | —      | —     | TWIDTH |       |        | 523              |
| 3D17h         | SPI1STATUS | TXWE          | —      | TXBE    | —      | RXRE  | CLRBFB | —     | RXBF   | 527              |
| 3D16h         | SPI1CON2   | BUSY          | SSFLT  | —       | —      | —     | SSET   | TXR   | RXR    | 526              |
| 3D15h         | SPI1CON1   | SMP           | CKE    | CKP     | FST    | —     | SSP    | SDIP  | SDOP   | 525              |
| 3D14h         | SPI1CON0   | EN            | —      | —       | —      | —     | LSBF   | MST   | BMODE  | 524              |
| 3D13h         | SPI1TCNTH  | —             | —      | —       | —      | —     | TCNTH  |       |        | 523              |
| 3D12h         | SPI1TCNTL  | TCNTL         |        |         |        |       |        |       | —      | 522              |
| 3D11h         | SPI1TXB    | TXB           |        |         |        |       |        |       | —      | 528              |
| 3D10h         | SPI1RXB    | RXB           |        |         |        |       |        |       | —      | 527              |
| 3D0Fh - 3CFFh | —          | Unimplemented |        |         |        |       |        |       | —      | —                |
| 3CFEh         | MD1CARH    | —             | —      | —       | CH     |       |        |       |        | 458              |
| 3CFDh         | MD1CARL    | —             | —      | —       | CL     |       |        |       |        | 458              |

**Legend:** x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

**Note 1:** Not present in LF devices.



# PIC18(L)F25/26K83

**TABLE 43-1: REGISTER FILE SUMMARY FOR PIC18(L)F25/26K83 DEVICES (CONTINUED)**

| Addr          | Name     | Bit 7         | Bit 6 | Bit 5 | Bit 4  | Bit 3   | Bit 2   | Bit 1   | Bit 0   | Register on page |
|---------------|----------|---------------|-------|-------|--------|---------|---------|---------|---------|------------------|
| 3CFCh         | MD1SRC   | —             | —     | —     | MS     |         |         |         |         | 459              |
| 3CFBh         | MD1CON1  | —             | —     | CHPOL | CHSYNC | —       | —       | CLPOL   | CLSYNC  | 457              |
| 3CFAh         | MD1CON0  | EN            | —     | OUT   | OPOL   | —       | —       | —       | BIT     | 456              |
| 3CF9h - 3CE7h | —        | Unimplemented |       |       |        |         |         |         |         | —                |
| 3CE6h         | CLKRCLK  | —             | —     | —     | —      | CLK     |         |         |         | 104              |
| 3CE5h         | CLKRCON  | EN            | —     | —     | DC     |         | DIV     |         |         | 103              |
| 3CE4h - 3C7Fh | —        | Unimplemented |       |       |        |         |         |         |         | —                |
| 3C7Eh         | CLCDATA0 | —             | —     | —     | —      | CLC4OUT | CLC3OUT | CLC2OUT | CLC1OUT | 434              |
| 3C7Dh         | CLC1GLS3 | G4D4T         | G4D4N | G4D3T | G4D3N  | G4D2T   | G4D2N   | G4D1T   | G4D1N   | 433              |
| 3C7Ch         | CLC1GLS2 | G3D4T         | G3D4N | G3D3T | G3D3N  | G3D2T   | G3D2N   | G3D1T   | G3D1N   | 432              |
| 3C7Bh         | CLC1GLS1 | G2D4T         | G2D4N | G2D3T | G2D3N  | G2D2T   | G2D2N   | G2D1T   | G2D1N   | 431              |
| 3C7Ah         | CLC1GLS0 | G1D4T         | G1D4N | G1D3T | G1D3N  | G1D2T   | G1D2N   | G1D1T   | G1D1N   | 430              |
| 3C79h         | CLC1SEL3 | D4S           |       |       |        |         |         |         |         | 429              |
| 3C78h         | CLC1SEL2 | D3S           |       |       |        |         |         |         |         | 429              |
| 3C77h         | CLC1SEL1 | D2S           |       |       |        |         |         |         |         | 429              |
| 3C76h         | CLC1SEL0 | D1S           |       |       |        |         |         |         |         | 429              |
| 3C75h         | CLC1POL  | POL           | —     | —     | —      | G4POL   | G3POL   | G2POL   | G1POL   | 428              |
| 3C74h         | CLC1CON  | EN            | OE    | OUT   | INTP   | INTN    | MODE    |         |         | 427              |
| 3C73h         | CLC2GLS3 | G4D4T         | G4D4N | G4D3T | G4D3N  | G4D2T   | G4D2N   | G4D1T   | G4D1N   | 433              |
| 3C72h         | CLC2GLS2 | G3D4T         | G3D4N | G3D3T | G3D3N  | G3D2T   | G3D2N   | G3D1T   | G3D1N   | 432              |
| 3C71h         | CLC2GLS1 | G2D4T         | G2D4N | G2D3T | G2D3N  | G2D2T   | G2D2N   | G2D1T   | G2D1N   | 431              |
| 3C70h         | CLC2GLS0 | G1D4T         | G1D4N | G1D3T | G1D3N  | G1D2T   | G1D2N   | G1D1T   | G1D1N   | 430              |
| 3C6Fh         | CLC2SEL3 | D4S           |       |       |        |         |         |         |         | 429              |
| 3C6Eh         | CLC2SEL2 | D3S           |       |       |        |         |         |         |         | 429              |
| 3C6Dh         | CLC2SEL1 | D2S           |       |       |        |         |         |         |         | 429              |
| 3C6Ch         | CLC2SEL0 | D1S           |       |       |        |         |         |         |         | 429              |
| 3C6Bh         | CLC2POL  | POL           | —     | —     | —      | G4POL   | G3POL   | G2POL   | G1POL   | 428              |
| 3C6Ah         | CLC2CON  | EN            | OE    | OUT   | INTP   | INTN    | MODE    |         |         | 427              |
| 3C69h         | CLC3GLS3 | G4D4T         | G4D4N | G4D3T | G4D3N  | G4D2T   | G4D2N   | G4D1T   | G4D1N   | 433              |
| 3C68h         | CLC3GLS2 | G3D4T         | G3D4N | G3D3T | G3D3N  | G3D2T   | G3D2N   | G3D1T   | G3D1N   | 432              |
| 3C67h         | CLC3GLS1 | G2D4T         | G2D4N | G2D3T | G2D3N  | G2D2T   | G2D2N   | G2D1T   | G2D1N   | 431              |
| 3C66h         | CLC3GLS0 | G1D4T         | G1D4N | G1D3T | G1D3N  | G1D2T   | G1D2N   | G1D1T   | G1D1N   | 430              |
| 3C65h         | CLC3SEL3 | D4S           |       |       |        |         |         |         |         | 429              |
| 3C64h         | CLC3SEL2 | D3S           |       |       |        |         |         |         |         | 429              |
| 3C63h         | CLC3SEL1 | D2S           |       |       |        |         |         |         |         | 429              |
| 3C62h         | CLC3SEL0 | D1S           |       |       |        |         |         |         |         | 430              |
| 3C61h         | CLC3POL  | POL           | —     | —     | —      | G4POL   | G3POL   | G2POL   | G1POL   | 428              |
| 3C60h         | CLC3CON  | EN            | OE    | OUT   | INTP   | INTN    | MODE    |         |         | 427              |
| 3C5Fh         | CLC4GLS3 | G4D4T         | G4D4N | G4D3T | G4D3N  | G4D2T   | G4D2N   | G4D1T   | G4D1N   | 433              |
| 3C5Eh         | CLC4GLS2 | G3D4T         | G3D4N | G3D3T | G3D3N  | G3D2T   | G3D2N   | G3D1T   | G3D1N   | 432              |
| 3C5Dh         | CLC4GLS1 | G2D4T         | G2D4N | G2D3T | G2D3N  | G2D2T   | G2D2N   | G2D1T   | G2D1N   | 431              |
| 3C5Ch         | CLC4GLS0 | G1D4T         | G1D4N | G1D3T | G1D3N  | G1D2T   | G1D2N   | G1D1T   | G1D1N   | 430              |
| 3C5Bh         | CLC4SEL3 | D4S           |       |       |        |         |         |         |         | 429              |
| 3C5Ah         | CLC4SEL2 | D3S           |       |       |        |         |         |         |         | 429              |
| 3C59h         | CLC4SEL1 | D2S           |       |       |        |         |         |         |         | 429              |
| 3C58h         | CLC4SEL0 | D1S           |       |       |        |         |         |         |         | 430              |
| 3C57h         | CLC4POL  | POL           | —     | —     | —      | G4POL   | G3POL   | G2POL   | G1POL   | 428              |
| 3C56h         | CLC4CON  | EN            | OE    | OUT   | INTP   | INTN    | MODE    |         |         | 427              |

**Legend:** x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

**Note 1:** Not present in LF devices.

# PIC18(L)F25/26K83

**TABLE 43-1: REGISTER FILE SUMMARY FOR PIC18(L)F25/26K83 DEVICES (CONTINUED)**

| Addr          | Name      | Bit 7         | Bit 6  | Bit 5 | Bit 4 | Bit 3 | Bit 2  | Bit 1 | Bit 0 | Register on page |
|---------------|-----------|---------------|--------|-------|-------|-------|--------|-------|-------|------------------|
| 3C55h - 3C00h | —         | Unimplemented |        |       |       |       |        |       |       | —                |
| 3BFFh         | DMA1SIRQ  | SIRQ          |        |       |       |       |        |       |       | 247              |
| 3BFEh         | DMA1AIRQ  | AIRQ          |        |       |       |       |        |       |       | 247              |
| 3BFDh         | DMA1CON1  | DMODE         |        | DSTP  | SMR   |       | SMODE  |       | SSTP  | 240              |
| 3BFCCh        | DMA1CON0  | EN            | SIRQEN | DGO   | —     | —     | AIRQEN | —     | XIP   | 239              |
| 3BFBh         | DMA1SSAU  | —             | —      | SSA   |       |       |        |       |       | 242              |
| 3BFAh         | DMA1SSAH  | SSA           |        |       |       |       |        |       |       | 241              |
| 3BF9h         | DMA1SSAL  | SSA           |        |       |       |       |        |       |       | 241              |
| 3BF8h         | DMA1SSZH  | —             | —      | —     | —     | SSZ   |        |       |       | 243              |
| 3BF7h         | DMA1SSZL  | SSZ           |        |       |       |       |        |       |       | 243              |
| 3BF6h         | DMA1SPTRU | —             | —      | SPTR  |       |       |        |       |       | 243              |
| 3BF5h         | DMA1SPTRH | SPTR          |        |       |       |       |        |       |       | 242              |
| 3BF4h         | DMA1SPTRL | SPTR          |        |       |       |       |        |       |       | 242              |
| 3BF3h         | DMA1SCNTH | —             | —      | —     | —     | SCNT  |        |       |       | 244              |
| 3BF2h         | DMA1SCNTL | SCNT          |        |       |       |       |        |       |       | 244              |
| 3BF1h         | DMA1DSAH  | DSA           |        |       |       |       |        |       |       | 245              |
| 3BF0h         | DMA1DSAL  | SSA           |        |       |       |       |        |       |       | 244              |
| 3BEFh         | DMA1DSZH  | —             | —      | —     | —     | DSZ   |        |       |       | 246              |
| 3BEEh         | DMA1DSZL  | DSZ           |        |       |       |       |        |       |       | 246              |
| 3BEDh         | DMA1DPTRH | DPTR          |        |       |       |       |        |       |       | 245              |
| 3BECCh        | DMA1DPTRL | DPTR          |        |       |       |       |        |       |       | 245              |
| 3BEBh         | DMA1DCNTH | —             | —      | —     | —     | DCNT  |        |       |       | 244              |
| 3BEAh         | DMA1DCNTL | DCNT          |        |       |       |       |        |       |       | 246              |
| 3BE9h         | DMA1BUF   | BUF           |        |       |       |       |        |       |       | 241              |
| 3BE8h - 3BE0h | —         | Unimplemented |        |       |       |       |        |       |       | —                |
| 3BDFh         | DMA2SIRQ  | —             | SIRQ   |       |       |       |        |       |       | 247              |
| 3BDEh         | DMA2AIRQ  | —             | AIRQ   |       |       |       |        |       |       | 247              |
| 3BDDh         | DMA2CON1  | DMODE         |        | DSTP  | SMR   |       | SMODE  |       | SSTP  | 240              |
| 3BDCh         | DMA2CON0  | EN            | SIRQEN | DGO   | —     | —     | AIRQEN | —     | XIP   | 239              |
| 3BDBh         | DMA2SSAU  | —             | —      | SSA   |       |       |        |       |       | 242              |
| 3BDAh         | DMA2SSAH  | SSA           |        |       |       |       |        |       |       | 241              |
| 3BD9h         | DMA2SSAL  | SSA           |        |       |       |       |        |       |       | 241              |
| 3BD8h         | DMA2SSZH  | —             | —      | —     | —     | SSZ   |        |       |       | 243              |
| 3BD7h         | DMA2SSZL  | SSZ           |        |       |       |       |        |       |       | 243              |
| 3BD6h         | DMA2SPTRU | —             | —      | SPTR  |       |       |        |       |       | 243              |
| 3BD5h         | DMA2SPTRH | SPTR          |        |       |       |       |        |       |       | 242              |
| 3BD4h         | DMA2SPTRL | SPTR          |        |       |       |       |        |       |       | 242              |
| 3BD3h         | DMA2SCNTH | —             | —      | —     | —     | SCNT  |        |       |       | 244              |
| 3BD2h         | DMA2SCNTL | SCNT          |        |       |       |       |        |       |       | 244              |
| 3BD1h         | DMA2DSAH  | DSA           |        |       |       |       |        |       |       | 245              |
| 3BD0h         | DMA2DSAL  | SSA           |        |       |       |       |        |       |       | 244              |
| 3BCFh         | DMA2DSZH  | —             | —      | —     | —     | DSZ   |        |       |       | 246              |
| 3BCEh         | DMA2DSZL  | DSZ           |        |       |       |       |        |       |       | 246              |
| 3BCDh         | DMA2DPTRH | DPTR          |        |       |       |       |        |       |       | 245              |
| 3BCCh         | DMA2DPTRL | DPTR          |        |       |       |       |        |       |       | 245              |
| 3BCBh         | DMA2DCNTH | —             | —      | —     | —     | DCNT  |        |       |       | 244              |
| 3BCAh         | DMA2DCNTL | DCNT          |        |       |       |       |        |       |       | 246              |
| 3BC9h         | DMA2BUF   | BUF           |        |       |       |       |        |       |       | 241              |

**Legend:** x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

**Note 1:** Not present in LF devices.

# PIC18(L)F25/26K83

**TABLE 43-1: REGISTER FILE SUMMARY FOR PIC18(L)F25/26K83 DEVICES (CONTINUED)**

| Addr          | Name       | Bit 7         | Bit 6 | Bit 5 | Bit 4      | Bit 3 | Bit 2 | Bit 1 | Bit 0     | Register on page |
|---------------|------------|---------------|-------|-------|------------|-------|-------|-------|-----------|------------------|
| 3BC8h - 3AEEh | —          | Unimplemented |       |       |            |       |       |       |           | —                |
| 3AEDh         | CANRXPPS   | —             | —     | —     | CANRXPPS   |       |       |       |           | 265              |
| 3AEC          | —          | Unimplemented |       |       |            |       |       |       |           | —                |
| 3AEBh         | U2CTSPPS   | —             | —     | —     | U2CTSPPS   |       |       |       |           | 265              |
| 3AEA          | U2RXPPS    | —             | —     | —     | U2RXPPS    |       |       |       |           | 265              |
| 3AE9h         | —          | Unimplemented |       |       |            |       |       |       |           | —                |
| 3AE8h         | U1CTSPPS   | —             | —     | —     | U1CTSPPS   |       |       |       |           | 265              |
| 3AE7h         | U1RXPPS    | —             | —     | —     | U1RXPPS    |       |       |       |           | 265              |
| 3AE6h         | I2C2SDAPPS | —             | —     | —     | I2C2SDAPPS |       |       |       |           | 265              |
| 3AE5h         | I2C2SCLPPS | —             | —     | —     | I2C2SCLPPS |       |       |       |           | 265              |
| 3AE4h         | I2C1SDAPPS | —             | —     | —     | I2C1SDAPPS |       |       |       |           | 265              |
| 3AE3h         | I2C1SCLPPS | —             | —     | —     | I2C1SCLPPS |       |       |       |           | 265              |
| 3AE2h         | SPI1SSPPS  | —             | —     | —     | SPI1SSPPS  |       |       |       |           | 265              |
| 3AE1h         | SPI1SDIPPS | —             | —     | —     | SPI1SDIPPS |       |       |       |           | 265              |
| 3AE0h         | SPI1SCKPPS | —             | —     | —     | SPI1SCKPPS |       |       |       |           | 265              |
| 3ADFh         | ADACTPPS   | —             | —     | —     | ADACTPPS   |       |       |       |           | 265              |
| 3ADEh         | CLCIN3PPS  | —             | —     | —     | CLCIN3PPS  |       |       |       |           | 265              |
| 3ADDh         | CLCIN2PPS  | —             | —     | —     | CLCIN2PPS  |       |       |       |           | 265              |
| 3ADCh         | CLCIN1PPS  | —             | —     | —     | CLCIN1PPS  |       |       |       |           | 265              |
| 3ADBh         | CLCIN0PPS  | —             | —     | —     | CLCIN0PPS  |       |       |       |           | 265              |
| 3ADAh         | MD1SRCPPS  | —             | —     | —     | MD1SRCPPS  |       |       |       |           | 265              |
| 3AD9h         | MD1CARHPPS | —             | —     | —     | MD1CARHPPS |       |       |       |           | 265              |
| 3AD8h         | MD1CARLPPS | —             | —     | —     | MD1CARLPPS |       |       |       |           | 265              |
| 3AD7h         | CWG3INPPS  | —             | —     | —     | CWG3INPPS  |       |       |       |           | 265              |
| 3AD6h         | CWG2INPPS  | —             | —     | —     | CWG2INPPS  |       |       |       |           | 265              |
| 3AD5h         | CWG1INPPS  | —             | —     | —     | CWG1INPPS  |       |       |       |           | 265              |
| 3AD4h         | SMT2SIGPPS | —             | —     | —     | SMT2SIGPPS |       |       |       |           | 265              |
| 3AD3h         | SMT2WINPPS | —             | —     | —     | SMT2WINPPS |       |       |       |           | 265              |
| 3AD2h         | SMT1SIGPPS | —             | —     | —     | SMT1SIGPPS |       |       |       |           | 265              |
| 3AD1h         | SMT1WINPPS | —             | —     | —     | SMT1WINPPS |       |       |       |           | 265              |
| 3AD0h         | CCP4PPS    | —             | —     | —     | CCP4PPS    |       |       |       |           | 265              |
| 3ACFh         | CCP3PPS    | —             | —     | —     | CCP3PPS    |       |       |       |           | 265              |
| 3ACEh         | CCP2PPS    | —             | —     | —     | CCP2PPS    |       |       |       |           | 265              |
| 3ACDh         | CCP1PPS    | —             | —     | —     | CCP1PPS    |       |       |       |           | 265              |
| 3ACCh         | T6INPPS    | —             | —     | —     | T6INPPS    |       |       |       |           | 265              |
| 3ACBh         | T4INPPS    | —             | —     | —     | T4INPPS    |       |       |       |           | 265              |
| 3ACAh         | T2INPPS    | —             | —     | —     | T2INPPS    |       |       |       |           | 265              |
| 3AC9h         | T5GPPS     | —             | —     | —     | T5GPPS     |       |       |       |           | 265              |
| 3AC8h         | T5CLKIPPS  | —             | —     | —     | T5CLKIPPS  |       |       |       |           | 265              |
| 3AC7h         | T3GPPS     | —             | —     | —     | T3GPPS     |       |       |       |           | 265              |
| 3AC6h         | T3CLKIPPS  | —             | —     | —     | T3CLKIPPS  |       |       |       |           | 265              |
| 3AC5h         | T1GPPS     | —             | —     | —     | T1GPPS     |       |       |       |           | 265              |
| 3AC4h         | T1CKIPPS   | —             | —     | —     | T1CKIPPS   |       |       |       |           | 265              |
| 3AC3h         | T0CKIPPS   | —             | —     | —     | T0CKIPPS   |       |       |       |           | 265              |
| 3AC2h         | INT2PPS    | —             | —     | —     | INT2PPS    |       |       |       |           | 265              |
| 3AC1h         | INT1PPS    | —             | —     | —     | INT1PPS    |       |       |       |           | 265              |
| 3AC0h         | INT0PPS    | —             | —     | —     | INT0PPS    |       |       |       |           | 265              |
| 3ABFh         | PPSLOCK    | —             | —     | —     | —          | —     | —     | —     | PPSLOCKED | 269              |

**Legend:** x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

**Note 1:** Not present in LF devices.

# PIC18(L)F25/26K83

**TABLE 43-1: REGISTER FILE SUMMARY FOR PIC18(L)F25/26K83 DEVICES (CONTINUED)**

| Addr          | Name    | Bit 7         | Bit 6   | Bit 5   | Bit 4   | Bit 3   | Bit 2   | Bit 1   | Bit 0   | Register on page |
|---------------|---------|---------------|---------|---------|---------|---------|---------|---------|---------|------------------|
| 3ABEh - 3A88h | —       | Unimplemented |         |         |         |         |         |         |         | —                |
| 3A87h         | IOCEF   | —             | —       | —       | —       | IOCEF3  | —       | —       | —       | 273              |
| 3A86h         | IOCEN   | —             | —       | —       | —       | IOCEN3  | —       | —       | —       | 273              |
| 3A85h         | IOCEP   | —             | —       | —       | —       | IOCEP3  | —       | —       | —       | 273              |
| 3A84h         | INLVLE  | —             | —       | —       | —       | INLVLE3 | —       | —       | —       | 260              |
| 3A83h         | —       | Unimplemented |         |         |         |         |         |         |         | —                |
| 3A82h         | —       | Unimplemented |         |         |         |         |         |         |         | —                |
| 3A81h         | WPUE    | —             | —       | —       | —       | WPUE3   | —       | —       | —       | 257              |
| 3A80h - 3A6Ch | —       | Unimplemented |         |         |         |         |         |         |         | —                |
| 3A6Bh         | RC4I2C  | —             | SLEW    | PU      | —       | —       | —       | TH      | —       | 253              |
| 3A6Ah         | RC3I2C  | —             | SLEW    | PU      | —       | —       | —       | TH      | —       | 253              |
| 3A69h         | —       | Unimplemented |         |         |         |         |         |         |         | —                |
| 3A68h         | —       | Unimplemented |         |         |         |         |         |         |         | —                |
| 3A67h         | IOCCF   | IOCCF7        | IOCCF6  | IOCCF5  | IOCCF4  | IOCCF3  | IOCCF2  | IOCCF1  | IOCCF0  | 273              |
| 3A66h         | IOCCN   | IOCCN7        | IOCCN6  | IOCCN5  | IOCCN4  | IOCCN3  | IOCCN2  | IOCCN1  | IOCCN0  | 273              |
| 3A65h         | IOCCP   | IOCCP7        | IOCCP6  | IOCCP5  | IOCCP4  | IOCCP3  | IOCCP2  | IOCCP1  | IOCCP0  | 273              |
| 3A64h         | INLVLC  | INLVLC7       | INLVLC6 | INLVLC5 | INLVLC4 | INLVLC3 | INLVLC2 | INLVLC1 | INLVLC0 | 260              |
| 3A63h         | SLRCONC | SLRC7         | SLRC6   | SLRC5   | SLRC4   | SLRC3   | SLRC2   | SLRC1   | SLRC0   | 259              |
| 3A62h         | ODCONC  | ODCC7         | ODCC6   | ODCC5   | ODCC4   | ODCC3   | ODCC2   | ODCC1   | ODCC0   | 258              |
| 3A61h         | WPUC    | WPUC7         | WPUC6   | WPUC5   | WPUC4   | WPUC3   | WPUC2   | WPUC1   | WPUC0   | 257              |
| 3A60h         | ANSELC  | ANSELC7       | ANSELC6 | ANSELC5 | ANSELC4 | ANSELC3 | ANSELC2 | ANSELC1 | ANSELC0 | 256              |
| 3A5Fh - 3A5Ch | —       | Unimplemented |         |         |         |         |         |         |         | —                |
| 3A5Bh         | RB2I2C  | —             | SLEW    | PU      | —       | —       | —       | TH      | —       | 253              |
| 3A5Ah         | RB1I2C  | —             | SLEW    | PU      | —       | —       | —       | TH      | —       | 253              |
| 3A59h         | —       | Unimplemented |         |         |         |         |         |         |         | —                |
| 3A58h         | —       | Unimplemented |         |         |         |         |         |         |         | —                |
| 3A57h         | IOCBF   | IOCBF7        | IOCBF6  | IOCBF5  | IOCBF4  | IOCBF3  | IOCBF2  | IOCBF1  | IOCBF0  | 273              |
| 3A56h         | IOCBN   | IOCBN7        | IOCBN6  | IOCBN5  | IOCBN4  | IOCBN3  | IOCBN2  | IOCBN1  | IOCBN0  | 273              |
| 3A55h         | IOCBP   | IOCBP7        | IOCBP6  | IOCBP5  | IOCBP4  | IOCBP3  | IOCBP2  | IOCBP1  | IOCBP0  | 273              |
| 3A54h         | INVLVB  | INVLVB7       | INVLVB6 | INVLVB5 | INVLVB4 | INVLVB3 | INVLVB2 | INVLVB1 | INVLVB0 | 260              |
| 3A53h         | SLRCONB | SLRB7         | SLRB6   | SLRB5   | SLRB4   | SLRB3   | SLRB2   | SLRB1   | SLRB0   | 259              |
| 3A52h         | ODCONB  | ODCB7         | ODCB6   | ODCB5   | ODCB4   | ODCB3   | ODCB2   | ODCB1   | ODCB0   | 258              |
| 3A51h         | WPUB    | WPUB7         | WPUB6   | WPUB5   | WPUB4   | WPUB3   | WPUB2   | WPUB1   | WPUB0   | 257              |
| 3A50h         | ANSELB  | ANSELB7       | ANSELB6 | ANSELB5 | ANSELB4 | ANSELB3 | ANSELB2 | ANSELB1 | ANSELB0 | 256              |
| 3A4Fh - 3A48h | —       | Unimplemented |         |         |         |         |         |         |         | —                |
| 3A47h         | IOCAF   | IOCAF7        | IOCAF6  | IOCAF5  | IOCAF4  | IOCAF3  | IOCAF2  | IOCAF1  | IOCAF0  | 273              |
| 3A46h         | IOCAN   | IOCAN7        | IOCAN6  | IOCAN5  | IOCAN4  | IOCAN3  | IOCAN2  | IOCAN1  | IOCAN0  | 273              |
| 3A45h         | IOCAP   | IOCAP7        | IOCAP6  | IOCAP5  | IOCAP4  | IOCAP3  | IOCAP2  | IOCAP1  | IOCAP0  | 273              |
| 3A44h         | INLVLA  | INLVLA7       | INLVLA6 | INLVLA5 | INLVLA4 | INLVLA3 | INLVLA2 | INLVLA1 | INLVLA0 | 260              |
| 3A43h         | SLRCONA | SLRA7         | SLRA6   | SLRA5   | SLRA4   | SLRA3   | SLRA2   | SLRA1   | SLRA0   | 259              |
| 3A42h         | ODCONA  | ODCA7         | ODCA6   | ODCA5   | ODCA4   | ODCA3   | ODCA2   | ODCA1   | ODCA0   | 258              |
| 3A41h         | WPUA    | WPUA7         | WPUA6   | WPUA5   | WPUA4   | WPUA3   | WPUA2   | WPUA1   | WPUA0   | 257              |
| 3A40h         | ANSELA  | ANSELA7       | ANSELA6 | ANSELA5 | ANSELA4 | ANSELA3 | ANSELA2 | ANSELA1 | ANSELA0 | 256              |
| 3A3Fh - 3A18h | —       | Unimplemented |         |         |         |         |         |         |         | —                |
| 3A17h         | RC7PPS  | —             | —       | RC7PPS5 | RC7PPS4 | RC7PPS3 | RC7PPS2 | RC7PPS1 | RC7PPS0 | 267              |
| 3A16h         | RC6PPS  | —             | —       | RC6PPS5 | RC6PPS4 | RC6PPS3 | RC6PPS2 | RC6PPS1 | RC6PPS0 | 267              |

**Legend:** x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

**Note 1:** Not present in LF devices.

# PIC18(L)F25/26K83

**TABLE 43-1: REGISTER FILE SUMMARY FOR PIC18(L)F25/26K83 DEVICES (CONTINUED)**

| Addr          | Name    | Bit 7         | Bit 6   | Bit 5   | Bit 4   | Bit 3   | Bit 2   | Bit 1   | Bit 0    | Register on page    |
|---------------|---------|---------------|---------|---------|---------|---------|---------|---------|----------|---------------------|
| 3A15h         | RC5PPS  | —             | —       | RC5PPS5 | RC5PPS4 | RC5PPS3 | RC5PPS2 | RC5PPS1 | RC5PPS0  | <a href="#">267</a> |
| 3A14h         | RC4PPS  | —             | —       | RC4PPS5 | RC4PPS4 | RC4PPS3 | RC4PPS2 | RC4PPS1 | RC4PPS0  | <a href="#">267</a> |
| 3A13h         | RC3PPS  | —             | —       | RC3PPS5 | RC3PPS4 | RC3PPS3 | RC3PPS2 | RC3PPS1 | RC3PPS0  | <a href="#">267</a> |
| 3A12h         | RC2PPS  | —             | —       | RC2PPS5 | RC2PPS4 | RC2PPS3 | RC2PPS2 | RC2PPS1 | RC2PPS0  | <a href="#">267</a> |
| 3A11h         | RC1PPS  | —             | —       | RC1PPS5 | RC1PPS4 | RC1PPS3 | RC1PPS2 | RC1PPS1 | RC1PPS0  | <a href="#">267</a> |
| 3A10h         | RC0PPS  | —             | —       | RC0PPS5 | RC0PPS4 | RC0PPS3 | RC0PPS2 | RC0PPS1 | RC0PPS0  | <a href="#">267</a> |
| 3A0Fh         | RB7PPS  | —             | —       | RB7PPS5 | RB7PPS4 | RB7PPS3 | RB7PPS2 | RB7PPS1 | RB7PPS0  | <a href="#">267</a> |
| 3A0Eh         | RB6PPS  | —             | —       | RB6PPS5 | RB6PPS4 | RB6PPS3 | RB6PPS2 | RB6PPS1 | RB6PPS0  | <a href="#">267</a> |
| 3A0Dh         | RB5PPS  | —             | —       | RB5PPS5 | RB5PPS4 | RB5PPS3 | RB5PPS2 | RB5PPS1 | RB5PPS0  | <a href="#">267</a> |
| 3A0Ch         | RB4PPS  | —             | —       | RB4PPS5 | RB4PPS4 | RB4PPS3 | RB4PPS2 | RB4PPS1 | RB4PPS0  | <a href="#">267</a> |
| 3A0Bh         | RB3PPS  | —             | —       | RB3PPS5 | RB3PPS4 | RB3PPS3 | RB3PPS2 | RB3PPS1 | RB3PPS0  | <a href="#">267</a> |
| 3A0Ah         | RB2PPS  | —             | —       | RB2PPS5 | RB2PPS4 | RB2PPS3 | RB2PPS2 | RB2PPS1 | RB2PPS0  | <a href="#">267</a> |
| 3A09h         | RB1PPS  | —             | —       | RB1PPS5 | RB1PPS4 | RB1PPS3 | RB1PPS2 | RB1PPS1 | RB1PPS0  | <a href="#">267</a> |
| 3A08h         | RB0PPS  | —             | —       | RB0PPS5 | RB0PPS4 | RB0PPS3 | RB0PPS2 | RB0PPS1 | RB0PPS0  | <a href="#">267</a> |
| 3A07h         | RA7PPS  | —             | —       | RA7PPS5 | RA7PPS4 | RA7PPS3 | RA7PPS2 | RA7PPS1 | RA7PPS0  | <a href="#">267</a> |
| 3A06h         | RA6PPS  | —             | —       | RA6PPS5 | RA6PPS4 | RA6PPS3 | RA6PPS2 | RA6PPS1 | RA6PPS0  | <a href="#">267</a> |
| 3A05h         | RA5PPS  | —             | —       | RA5PPS5 | RA5PPS4 | RA5PPS3 | RA5PPS2 | RA5PPS1 | RA5PPS0  | <a href="#">267</a> |
| 3A04h         | RA4PPS  | —             | —       | RA4PPS5 | RA4PPS4 | RA4PPS3 | RA4PPS2 | RA4PPS1 | RA4PPS0  | <a href="#">267</a> |
| 3A03h         | RA3PPS  | —             | —       | RA3PPS5 | RA3PPS4 | RA3PPS3 | RA3PPS2 | RA3PPS1 | RA3PPS0  | <a href="#">267</a> |
| 3A02h         | RA2PPS  | —             | —       | RA2PPS5 | RA2PPS4 | RA2PPS3 | RA2PPS2 | RA2PPS1 | RA2PPS0  | <a href="#">267</a> |
| 3A01h         | RA1PPS  | —             | —       | RA1PPS5 | RA1PPS4 | RA1PPS3 | RA1PPS2 | RA1PPS1 | RA1PPS0  | <a href="#">267</a> |
| 3A00h         | RA0PPS  | —             | —       | RA0PPS5 | RA0PPS4 | RA0PPS3 | RA0PPS2 | RA0PPS1 | RA0PPS0  | <a href="#">267</a> |
| 39FFh - 39F8h | —       | Unimplemented |         |         |         |         |         |         |          | —                   |
| 39F7h         | SCANPR  | —             | —       | —       | —       | —       | PR      |         |          | <a href="#">21</a>  |
| 39F6h - 39F5h | —       | Unimplemented |         |         |         |         |         |         |          | —                   |
| 39F4h         | DMA2PR  | —             | —       | —       | —       | —       | PR      |         |          | <a href="#">21</a>  |
| 39F3h         | DMA1PR  | —             | —       | —       | —       | —       | PR      |         |          | <a href="#">20</a>  |
| 39F2h         | MAINPR  | —             | —       | —       | —       | —       | PR      |         |          | <a href="#">20</a>  |
| 39F1h         | ISRPR   | —             | —       | —       | —       | —       | PR      |         |          | <a href="#">20</a>  |
| 39F0h         | —       | Unimplemented |         |         |         |         |         |         |          | —                   |
| 39EFh         | PRLOCK  | —             | —       | —       | —       | —       | —       | —       | PRLOCKED | <a href="#">21</a>  |
| 39EEh - 39E7h | —       | Unimplemented |         |         |         |         |         |         |          | —                   |
| 39E6h         | NVMCON2 | NVMCON2       |         |         |         |         |         |         |          | <a href="#">201</a> |
| 39E5h         | NVMCON1 | REG           |         | —       | FREE    | WRERR   | WREN    | WR      | RD       | <a href="#">200</a> |
| 39E4h         | —       | Unimplemented |         |         |         |         |         |         |          | —                   |
| 39E3h         | NVMDAT  | DAT           |         |         |         |         |         |         |          | <a href="#">202</a> |
| 39E2h         | —       | Unimplemented |         |         |         |         |         |         |          | —                   |
| 39E1h         | —       | Unimplemented |         |         |         |         |         |         |          | —                   |
| 39E0h         | NVMADRL | ADR           |         |         |         |         |         |         |          | <a href="#">201</a> |
| 39DFh         | OSCFRQ  | —             | —       | —       | —       | FRQ     |         |         |          | <a href="#">97</a>  |
| 39DEh         | OSCTUNE | —             | —       | TUN     |         |         |         |         |          | <a href="#">98</a>  |
| 39DDh         | OSCEN   | EXTOEN        | HFOEN   | MFOEN   | LFOEN   | SOSCEN  | ADOEN   | —       | —        | <a href="#">99</a>  |
| 39DCh         | OSCSTAT | EXTOR         | HFOR    | MFOR    | LFOR    | SOR     | ADOR    | —       | PLL      | <a href="#">96</a>  |
| 39DBh         | OSCCON3 | CSWHOLD       | SOSCPWR | —       | ORDY    | NOSCR   | —       | —       | —        | <a href="#">95</a>  |
| 39DAh         | OSCCON2 | —             | COSC    |         |         | CDIV    |         |         |          | <a href="#">95</a>  |
| 39D9h         | OSCCON1 | —             | NOSC    |         |         | NDIV    |         |         |          | <a href="#">94</a>  |
| 39D8h         | CPUDOZE | IDLEN         | DOZEN   | ROI     | DOE     | —       | DOZE    |         |          | <a href="#">167</a> |

**Legend:** x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

**Note 1:** Not present in LF devices.

# PIC18(L)F25/26K83

**TABLE 43-1: REGISTER FILE SUMMARY FOR PIC18(L)F25/26K83 DEVICES (CONTINUED)**

| Addr          | Name                   | Bit 7         | Bit 6     | Bit 5      | Bit 4             | Bit 3     | Bit 2     | Bit 1             | Bit 0             | Register on page |
|---------------|------------------------|---------------|-----------|------------|-------------------|-----------|-----------|-------------------|-------------------|------------------|
| 39D7h - 39D2h | —                      | Unimplemented |           |            |                   |           |           |                   |                   | —                |
| 39D1h         | VREGCON <sup>(1)</sup> | —             | —         | —          | —                 | —         | —         | VREGPM            | —                 | 166              |
| 39D0h         | BORCON                 | SBORN         | —         | —          | —                 | —         | —         | —                 | BORRDY            | 75               |
| 39CFh - 39C8h | —                      | Unimplemented |           |            |                   |           |           |                   |                   | —                |
| 39C7h         | PMD7                   | CANMD         | —         | —          | —                 | —         | —         | DMA2MD            | DMA1MD            | 283              |
| 39C6h         | PMD6                   | —             | SMT2MD    | SMT1MD     | CLC4MD            | CLC3MD    | CLC2MD    | CLC1MD            | DSMMD             | 282              |
| 39C5h         | PMD5                   | —             | —         | U2MD       | U1MD              | —         | SPI1MD    | I2C2MD            | I2C1MD            | 281              |
| 39C4h         | PMD4                   | CWG3MD        | CWG2MD    | CWG1MD     | —                 | —         | —         | —                 | —                 | 280              |
| 39C3h         | PMD3                   | PWM8MD        | PWM7MD    | PWM6MD     | PWM5MD            | CCP4MD    | CCP3MD    | CCP2MD            | CCP1MD            | 279              |
| 39C2h         | PMD2                   | —             | DACMD     | ADCMD      | —                 | —         | CMP2MD    | CMP1MD            | ZCDMD             | 278              |
| 39C1h         | PMD1                   | NCO1MD        | TMR6MD    | TMR5MD     | TMR4MD            | TMR3MD    | TMR2MD    | TMR1MD            | TMR0MD            | 277              |
| 39C0h         | PMD0                   | SYSCMD        | FVRMD     | HLVDM      | CRCMD             | SCANMD    | NVMM      | CLKRMD            | IOCMD             | 276              |
| 39BFh - 39AAh | —                      | Unimplemented |           |            |                   |           |           |                   |                   | —                |
| 39A9h         | PIR9                   | —             | CLC4IF    | CCP4IF     | CLC3IF            | CWG3IF    | CCP3IF    | TMR6IF            | TMR5IF            | 136              |
| 39A8h         | PIR8                   | TMR5IF        | INT2IF    | CLC2IF     | CWG2IF            | CCP2IF    | TMR4IF    | TMR3GIF           | TMR3IF            | 135              |
| 39A7h         | PIR7                   | U2IF          | U2EIF     | U2TXIF     | U2RXIF            | I2C2EIF   | I2C2IF    | I2C2TXIF          | I2C2RXIF          | 134              |
| 39A6h         | PIR6                   | DMA2AIF       | DMA2ORIF  | DMA2DCNTIF | DMA2SCNTIF        | SMT2PWAIF | SMT2PRAIF | SMT2IF            | C2IF              | 133              |
| 39A5h         | PIR5                   | IRXIF         | WAKIF     | ERRIF      | TXB2IF/<br>TXBnIF | TXB1IF    | TXB0IF    | RXB1IF/<br>RXBnIF | RXB0IF/<br>FIFOIF | 132              |
| 39A4h         | PIR4                   | INT1IF        | CLC1IF    | CWG1IF     | NCO1IF            | CCP1IF    | TMR2IF    | TMR1GIF           | TMR1IF            | 131              |
| 39A3h         | PIR3                   | TMR0IF        | U1IF      | U1EIF      | U1TXIF            | U1RXIF    | I2C1EIF   | I2C1IF            | I2C1TXIF          | 130              |
| 39A2h         | PIR2                   | I2C1RXIF      | SPI1IF    | SPI1TXIF   | SPI1RXIF          | DMA1AIF   | DMA1ORIF  | DMA1DCNTIF        | DMA1SCNTIF        | 128              |
| 39A1h         | PIR1                   | SMT1PWAIF     | SMT1PRAIF | SMT1IF     | C1IF              | ADTIF     | ADIF      | ZCDIF             | INT0IF            | 128              |
| 39A0h         | PIR0                   | IOCIF         | CRCIF     | SCANIF     | NVMIF             | CSWIF     | OSFIF     | HLVDIF            | SWIF              | 127              |
| 399Fh - 399Ah | —                      | Unimplemented |           |            |                   |           |           |                   |                   | —                |
| 3999h         | PIE9                   | —             | CLC4IE    | CCP4IE     | CLC3IE            | CWG3IE    | CCP3IE    | TMR6IE            | TMR5IE            | 146              |
| 3998h         | PIE8                   | TMR5IE        | INT2IE    | CLC2IE     | CWG2IE            | CCP2IE    | TMR4IE    | TMR3GIE           | TMR3IE            | 145              |
| 3997h         | PIE7                   | U2IE          | U2EIE     | U2TXIE     | U2RXIE            | I2C2EIE   | I2C2IE    | I2C2TXIE          | I2C2RXIE          | 144              |
| 3996h         | PIE6                   | DMA2AIE       | DMA2ORIE  | DMA2DCNTIE | DMA2SCNTIE        | SMT2PWAIE | SMT2PRAIE | SMT2IE            | C2IE              | 143              |
| 3995h         | PIE5                   | IRXIE         | WAKIE     | ERRIE      | TXB2IE/<br>TXBnIE | TXB1IE    | TXB0IE    | RXB1IE/<br>RXBnIE | RXB0IE/<br>FIFOIF | 142              |
| 3994h         | PIE4                   | INT1IE        | CLC1IE    | CWG1IE     | NCO1IE            | CCP1IE    | TMR2IE    | TMR1GIE           | TMR1IE            | 141              |
| 3993h         | PIE3                   | TMR0IE        | U1IE      | U1EIE      | U1TXIE            | U1RXIE    | I2C1EIE   | I2C1IE            | I2C1TXIE          | 140              |
| 3992h         | PIE2                   | I2C1RXIE      | SPI1IE    | SPI1TXIE   | SPI1RXIE          | DMA1AIE   | DMA1ORIE  | DMA1DCNTIE        | DMA1SCNTIE        | 139              |
| 3991h         | PIE1                   | SMT1PWAIE     | SMT1PRAIE | SMT1IE     | C1IE              | ADTIE     | ADIE      | ZCDIE             | INT0IE            | 138              |
| 3990h         | PIE0                   | IOCIE         | CRCIE     | SCANIE     | NVMIE             | CSWIE     | OSFIE     | HLVDIE            | SWIE              | 137              |
| 398Fh - 398Ah | —                      | Unimplemented |           |            |                   |           |           |                   |                   | —                |
| 3989h         | IPR9                   | —             | CLC4IP    | CCP4IP     | CLC3IP            | CWG3IP    | CCP3IP    | TMR6IP            | TMR5IP            | 156              |
| 3988h         | IPR8                   | TMR5IP        | INT2IP    | CLC2IP     | CWG2IP            | CCP2IP    | TMR4IP    | TMR3GIP           | TMR3IP            | 155              |
| 3987h         | IPR7                   | U2IP          | U2EIP     | U2TXIP     | U2RXIP            | I2C2EIP   | I2C2IP    | I2C2TXIP          | I2C2RXIP          | 154              |
| 3986h         | IPR6                   | DMA2AIP       | DMA2ORIP  | DMA2DCNTIP | DMA2SCNTIP        | SMT2PWAIP | SMT2PRAIP | SMT2IP            | C2IP              | 153              |
| 3985h         | IPR5                   | IRXIP         | WAKIP     | ERRIP      | TXB2IP/<br>TXBnIP | TXB1IP    | TXB0IP    | RXB1IP/<br>RXBnIP | RXB0IP/<br>FIFOIF | 152              |
| 3984h         | IPR4                   | INT1IP        | CLC1IP    | CWG1IP     | NCO1IP            | CCP1IP    | TMR2IP    | TMR1GIP           | TMR1IP            | 151              |
| 3983h         | IPR3                   | TMR0IP        | U1IP      | U1EIP      | U1TXIP            | U1RXIP    | I2C1EIP   | I2C1IP            | I2C1TXIP          | 150              |
| 3982h         | IPR2                   | I2C1RXIP      | SPI1IP    | SPI1TXIP   | SPI1RXIP          | DMA1AIP   | DMA1ORIP  | DMA1DCNTIP        | DMA1SCNTIP        | 149              |
| 3981h         | IPR1                   | SMT1PWAIP     | SMT1PRAIP | SMT1IP     | C1IP              | ADTIP     | ADIP      | ZCDIP             | INT0IP            | 148              |

**Legend:** x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

**Note 1:** Not present in LF devices.

# PIC18(L)F25/26K83

**TABLE 43-1: REGISTER FILE SUMMARY FOR PIC18(L)F25/26K83 DEVICES (CONTINUED)**

| Addr          | Name        | Bit 7         | Bit 6           | Bit 5           | Bit 4  | Bit 3  | Bit 2  | Bit 1   | Bit 0  | Register on page |
|---------------|-------------|---------------|-----------------|-----------------|--------|--------|--------|---------|--------|------------------|
| 3980h         | IPR0        | IOCIP         | CRCIP           | SCANIP          | NVMIP  | CSWIP  | OSFIP  | HLVDIP  | SWIP   | 147              |
| 397Fh - 397Eh | —           | Unimplemented |                 |                 |        |        |        |         |        | —                |
| 397Dh         | SCANTRIG    | —             | —               | —               | —      | TSEL   |        |         |        | 216              |
| 397Ch         | SCANCON0    | EN            | TRIGEN          | SGO             | —      | —      | MREG   | BURSTMD | BUSY   | 212              |
| 397Bh         | SCANHADRU   | —             | —               | HADR            |        |        |        |         |        | 214              |
| 397Ah         | SCANHADRH   | HADR          |                 |                 |        |        |        |         |        | 215              |
| 3979h         | SCANHADRL   | HADR          |                 |                 |        |        |        |         |        | 215              |
| 3978h         | SCANLADRU   | —             | —               | LADR            |        |        |        |         |        | 213              |
| 3977h         | SCANLADRH   | LADR          |                 |                 |        |        |        |         |        | 213              |
| 3976h         | SCANLADRL   | LADR          |                 |                 |        |        |        |         |        | 214              |
| 3975h - 396Ah | —           | Unimplemented |                 |                 |        |        |        |         |        | —                |
| 3969h         | CRCCON1     | DLEN          |                 |                 |        | PLEN   |        |         |        | 208              |
| 3968h         | CRCCON0     | EN            | CRCGO           | BUSY            | ACCM   | —      | —      | SHIFTM  | FULL   | 208              |
| 3967h         | CRCXORH     | X15           | X14             | X13             | X12    | X11    | X10    | X9      | X8     | 211              |
| 3966h         | CRCXORL     | X7            | X6              | X5              | X4     | X3     | X2     | X1      | —      | 211              |
| 3965h         | CRCSHIFTH   | SHFT15        | SHFT14          | SHFT13          | SHFT12 | SHFT11 | SHFT10 | SHFT9   | SHFT8  | 210              |
| 3964h         | CRCSHIFTL   | SHFT7         | SHFT6           | SHFT5           | SHFT4  | SHFT3  | SHFT2  | SHFT1   | SHFT0  | 210              |
| 3963h         | CRCACCH     | ACC15         | ACC14           | ACC13           | ACC12  | ACC11  | ACC10  | ACC9    | ACC8   | 209              |
| 3962h         | CRCACCL     | ACC7          | ACC6            | ACC5            | ACC4   | ACC3   | ACC2   | ACC1    | ACC0   | 210              |
| 3961h         | CRCDATH     | DATA15        | DATA14          | DATA13          | DATA12 | DATA11 | DATA10 | DATA9   | DATA8  | 209              |
| 3960h         | CRCDATL     | DATA7         | DATA6           | DATA5           | DATA4  | DATA3  | DATA2  | DATA1   | DATA0  | 209              |
| 395Fh         | WDTTMR      | WDTTMR        |                 |                 |        |        | STATE  | PSCNT   |        | 175              |
| 395Eh         | WDTPSH      | PSCNT         |                 |                 |        |        |        |         |        | 174              |
| 395Dh         | WDTPSL      | PSCNT         |                 |                 |        |        |        |         |        | 174              |
| 395Ch         | WDTCON1     | —             | WDTCS           |                 |        | —      | WINDOW |         |        | 173              |
| 395Bh         | WDTCON0     | —             | —               | WDTPS           |        |        |        |         | SEN    | 172              |
| 395Ah - 38A0h | —           | Unimplemented |                 |                 |        |        |        |         |        | —                |
| 389Fh         | IVTADU      | AD            |                 |                 |        |        |        |         |        | 158              |
| 389Eh         | IVTADH      | AD            |                 |                 |        |        |        |         |        | 158              |
| 389Dh         | IVTADL      | AD            |                 |                 |        |        |        |         |        | 158              |
| 389Ch - 3891h | —           | Unimplemented |                 |                 |        |        |        |         |        | —                |
| 3890h         | PRODH_SHAD  | PRODH         |                 |                 |        |        |        |         |        | 115              |
| 388Fh         | PRODL_SHAD  | PRODL         |                 |                 |        |        |        |         |        | 115              |
| 388Eh         | FSR2H_SHAD  | —             | —               | FSR2H           |        |        |        |         |        | 115              |
| 388Dh         | FSR2L_SHAD  | FSR2L         |                 |                 |        |        |        |         |        | 115              |
| 388Ch         | FSR1H_SHAD  | —             | —               | FSR1H           |        |        |        |         |        | 115              |
| 388Bh         | FSR1L_SHAD  | FSR1L         |                 |                 |        |        |        |         |        | 115              |
| 388Ah         | FSR0H_SHAD  | —             | —               | FSR0H           |        |        |        |         |        | 115              |
| 3889h         | FSR0L_SHAD  | FSR0L         |                 |                 |        |        |        |         |        | 115              |
| 3888h         | PCLATU_SHAD | —             | —               | —               | PCU    |        |        |         |        | 115              |
| 3887h         | PCLATH_SHAD | PCH           |                 |                 |        |        |        |         |        | 115              |
| 3886h         | BSR_SHAD    | —             | —               | BSR             |        |        |        |         |        | 115              |
| 3885h         | WREG_SHAD   | WREG          |                 |                 |        |        |        |         |        | 115              |
| 3884h         | STATUS_SHAD | —             | $\overline{TO}$ | $\overline{PD}$ | N      | OV     | Z      | DC      | C      | 115              |
| 3883h         | SHADCON     | —             | —               | —               | —      | —      | —      | —       | SHADLO | 159              |
| 3882h         | BSR_CSHAD   | —             | —               | BSR             |        |        |        |         |        | 47               |
| 3881h         | WREG_CSHAD  | WREG          |                 |                 |        |        |        |         |        | 47               |

**Legend:** x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

**Note 1:** Not present in LF devices.

# PIC18(L)F25/26K83

**TABLE 43-1: REGISTER FILE SUMMARY FOR PIC18(L)F25/26K83 DEVICES (CONTINUED)**

| Addr          | Name         | Bit 7         | Bit 6 | Bit 5  | Bit 4   | Bit 3   | Bit 2   | Bit 1   | Bit 0   | Register on page |
|---------------|--------------|---------------|-------|--------|---------|---------|---------|---------|---------|------------------|
| 3880h         | STATUS_CSHAD | —             | TO    | PD     | N       | OV      | Z       | DC      | C       | 47               |
| 387Fh - 3800h | —            | Unimplemented |       |        |         |         |         |         |         | —                |
| 37FFh         | CANCON_RO0   | CANCON_RO0    |       |        |         |         |         |         |         | 604              |
| 37FEh         | CANSTAT_RO0  | CANSTAT_RO0   |       |        |         |         |         |         |         | 605              |
| 37FDh         | RXB1D7       | RXB1D7        |       |        |         |         |         |         |         | 621              |
| 37FCh         | RXB1D6       | RXB1D6        |       |        |         |         |         |         |         | 621              |
| 37FBh         | RXB1D5       | RXB1D5        |       |        |         |         |         |         |         | 621              |
| 37FAh         | RXB1D4       | RXB1D4        |       |        |         |         |         |         |         | 621              |
| 37F9h         | RXB1D3       | RXB1D3        |       |        |         |         |         |         |         | 621              |
| 37F8h         | RXB1D2       | RXB1D2        |       |        |         |         |         |         |         | 621              |
| 37F7h         | RXB1D1       | RXB1D1        |       |        |         |         |         |         |         | 621              |
| 37F6h         | RXB1D0       | RXB1D0        |       |        |         |         |         |         |         | 621              |
| 37F5h         | RXB1DLC      | —             | RXRTR | RB1    | R0      | DLC3    | DLC2    | DLC1    | DLC0    | 621              |
| 37F4h         | RXB1EIDL     | EID7          | EID6  | EID5   | EID4    | EID3    | EID2    | EID1    | EID0    | 620              |
| 37F3h         | RXB1EIDH     | EID15         | EID14 | EID13  | EID12   | EID11   | EID10   | EID9    | EID8    | 620              |
| 37F2h         | RXB1SIDL     | SID2          | SID1  | SID0   | SRR     | EXID    | —       | EID17   | EID16   | 620              |
| 37F1h         | RXB1SIDH     | SID10         | SID9  | SID8   | SID7    | SID6    | SID5    | SID4    | SID3    | 619              |
| 37F0h         | RXB1CON      | RXFUL         | RXM1  | RXM0   | —       | RXRTRRO | FILHIT2 | FILHIT1 | FILHIT0 | 618              |
| 37F0h         | RXB1CON      | RXFUL         | RXM1  | RTRRO  | FILHIT4 | FILHIT3 | FILHIT2 | FILHIT1 | FILHIT0 | 618              |
| 37EFh         | CANCON_RO1   | CANCON_RO1    |       |        |         |         |         |         |         | 604              |
| 37EEh         | CANSTAT_RO1  | CANSTAT_RO1   |       |        |         |         |         |         |         | 605              |
| 37EDh         | TXB0D7       | TXB0D7        |       |        |         |         |         |         |         | 612              |
| 37ECh         | TXB0D6       | TXB0D6        |       |        |         |         |         |         |         | 612              |
| 37EBh         | TXB0D5       | TXB0D5        |       |        |         |         |         |         |         | 612              |
| 37EAh         | TXB0D4       | TXB0D4        |       |        |         |         |         |         |         | 612              |
| 37E9h         | TXB0D3       | TXB0D3        |       |        |         |         |         |         |         | 612              |
| 37E8h         | TXB0D2       | TXB0D2        |       |        |         |         |         |         |         | 612              |
| 37E7h         | TXB0D1       | TXB0D1        |       |        |         |         |         |         |         | 612              |
| 37E6h         | TXB0D0       | TXB0D0        |       |        |         |         |         |         |         | 612              |
| 37E5h         | TXB0DLC      | —             | TXRTR | —      | —       | DLC3    | DLC2    | DLC1    | DLC0    | 613              |
| 37E4h         | TXB0EIDL     | EID7          | EID6  | EID5   | EID4    | EID3    | EID2    | EID1    | EID0    | 612              |
| 37E3h         | TXB0EIDH     | EID15         | EID14 | EID13  | EID12   | EID11   | EID10   | EID9    | EID8    | 611              |
| 37E2h         | TXB0SIDL     | SID2          | SID1  | SID0   | —       | EXIDE   | —       | EID17   | EID16   | 611              |
| 37E1h         | TXB0SIDH     | SID10         | SID9  | SID8   | SID7    | SID6    | SID5    | SID4    | SID3    | 611              |
| 37E0h         | TXB0CON      | TXBIF         | TXABT | TXLARB | TXERR   | TXREQ   | —       | TXPRI1  | TXPRI0  | 610              |
| 37DFh         | CANCON_RO2   | CANCON_RO2    |       |        |         |         |         |         |         | 604              |
| 37DEh         | CANSTAT_RO2  | CANSTAT_RO2   |       |        |         |         |         |         |         | 605              |
| 37DDh         | TXB1D7       | TXB1D7        |       |        |         |         |         |         |         | 612              |
| 37DCh         | TXB1D6       | TXB1D6        |       |        |         |         |         |         |         | 612              |
| 37DBh         | TXB1D5       | TXB1D5        |       |        |         |         |         |         |         | 612              |
| 37DAh         | TXB1D4       | TXB1D4        |       |        |         |         |         |         |         | 612              |
| 37D9h         | TXB1D3       | TXB1D3        |       |        |         |         |         |         |         | 612              |
| 37D8h         | TXB1D2       | TXB1D2        |       |        |         |         |         |         |         | 612              |
| 37D7h         | TXB1D1       | TXB1D1        |       |        |         |         |         |         |         | 612              |
| 37D6h         | TXB1D0       | TXB1D0        |       |        |         |         |         |         |         | 612              |
| 37D5h         | TXB1DLC      | —             | TXRTR | —      | —       | DLC3    | DLC2    | DLC1    | DLC0    | 613              |
| 37D4h         | TXB1EIDL     | EID7          | EID6  | EID5   | EID4    | EID3    | EID2    | EID1    | EID0    | 612              |
| 37D3h         | TXB1EIDH     | EID15         | EID14 | EID13  | EID12   | EID11   | EID10   | EID9    | EID8    | 611              |

**Legend:** x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

**Note 1:** Not present in LF devices.



# PIC18(L)F25/26K83

**TABLE 43-1: REGISTER FILE SUMMARY FOR PIC18(L)F25/26K83 DEVICES (CONTINUED)**

| Addr  | Name        | Bit 7       | Bit 6 | Bit 5  | Bit 4 | Bit 3  | Bit 2 | Bit 1  | Bit 0  | Register on page    |
|-------|-------------|-------------|-------|--------|-------|--------|-------|--------|--------|---------------------|
| 37D2h | TXB1SIDL    | SID2        | SID1  | SID0   | —     | EXIDE  | —     | EID17  | EID16  | <a href="#">611</a> |
| 37D1h | TXB1SIDH    | SID10       | SID9  | SID8   | SID7  | SID6   | SID5  | SID4   | SID3   | <a href="#">611</a> |
| 37D0h | TXB1CON     | TXBIF       | TXABT | TXLARB | TXERR | TXREQ  | —     | TXPRI1 | TXPRI0 | <a href="#">610</a> |
| 37CFh | CANCON_R03  | CANCON_RO3  |       |        |       |        |       |        |        | <a href="#">604</a> |
| 37CEh | CANSTAT_R03 | CANSTAT_RO3 |       |        |       |        |       |        |        | <a href="#">605</a> |
| 37CDh | TXB2D7      | TXB2D7      |       |        |       |        |       |        |        | <a href="#">612</a> |
| 37CCh | TXB2D6      | TXB2D6      |       |        |       |        |       |        |        | <a href="#">612</a> |
| 37CBh | TXB2D5      | TXB2D5      |       |        |       |        |       |        |        | <a href="#">612</a> |
| 37CAh | TXB2D4      | TXB2D4      |       |        |       |        |       |        |        | <a href="#">612</a> |
| 37C9h | TXB2D3      | TXB2D3      |       |        |       |        |       |        |        | <a href="#">612</a> |
| 37C8h | TXB2D2      | TXB2D2      |       |        |       |        |       |        |        | <a href="#">612</a> |
| 37C7h | TXB2D1      | TXB2D1      |       |        |       |        |       |        |        | <a href="#">612</a> |
| 37C6h | TXB2D0      | TXB2D0      |       |        |       |        |       |        |        | <a href="#">612</a> |
| 37C5h | TXB2DLC     | —           | TXRTR | —      | —     | DLC3   | DLC2  | DLC1   | DLC0   | <a href="#">613</a> |
| 37C4h | TXB2EIDL    | EID7        | EID6  | EID5   | EID4  | EID3   | EID2  | EID1   | EID0   | <a href="#">612</a> |
| 37C3h | TXB2EIDH    | EID15       | EID14 | EID13  | EID12 | EID11  | EID10 | EID9   | EID8   | <a href="#">611</a> |
| 37C2h | TXB2SIDL    | SID2        | SID1  | SID0   | —     | EXIDE  | —     | EID17  | EID16  | <a href="#">611</a> |
| 37C1h | TXB2SIDH    | SID10       | SID9  | SID8   | SID7  | SID6   | SID5  | SID4   | SID3   | <a href="#">611</a> |
| 37C0h | TXB2CON     | TXBIF       | TXABT | TXLARB | TXERR | TXREQ  | —     | TXPRI1 | TXPRI0 | <a href="#">610</a> |
| 37BFh | RXM1EIDL    | EID7        | EID6  | EID5   | EID4  | EID3   | EID2  | EID1   | EID0   | <a href="#">634</a> |
| 37BEh | RXM1EIDH    | EID15       | EID14 | EID13  | EID12 | EID11  | EID10 | EID9   | EID8   | <a href="#">633</a> |
| 37BDh | RXM1SIDL    | SID2        | SID1  | SID0   | —     | EXIDEN | —     | EID17  | EID16  | <a href="#">633</a> |
| 37BCh | RXM1SIDH    | SID10       | SID9  | SID8   | SID7  | SID6   | SID5  | SID4   | SID3   | <a href="#">632</a> |
| 37BBh | RXM0EIDL    | EID7        | EID6  | EID5   | EID4  | EID3   | EID2  | EID1   | EID0   | <a href="#">634</a> |
| 37BAh | RXM0EIDH    | EID15       | EID14 | EID13  | EID12 | EID11  | EID10 | EID9   | EID8   | <a href="#">633</a> |
| 37B9h | RXM0SIDL    | SID2        | SID1  | SID0   | —     | EXIDEN | —     | EID17  | EID16  | <a href="#">632</a> |
| 37B8h | RXM0SIDH    | SID10       | SID9  | SID8   | SID7  | SID6   | SID5  | SID4   | SID3   | <a href="#">632</a> |
| 37B7h | RXF5EIDL    | EID7        | EID6  | EID5   | EID4  | EID3   | EID2  | EID1   | EID0   | <a href="#">632</a> |
| 37B6h | RXF5EIDH    | EID15       | EID14 | EID13  | EID12 | EID11  | EID10 | EID9   | EID8   | <a href="#">632</a> |
| 37B5h | RXF5SIDL    | SID2        | SID1  | SID0   | —     | EXIDEN | —     | EID17  | EID16  | <a href="#">631</a> |
| 37B4h | RXF5SIDH    | SID10       | SID9  | SID8   | SID7  | SID6   | SID5  | SID4   | SID3   | <a href="#">631</a> |
| 37B3h | RXF4EIDL    | EID7        | EID6  | EID5   | EID4  | EID3   | EID2  | EID1   | EID0   | <a href="#">632</a> |
| 37B2h | RXF4EIDH    | EID15       | EID14 | EID13  | EID12 | EID11  | EID10 | EID9   | EID8   | <a href="#">632</a> |
| 37B1h | RXF4SIDL    | SID2        | SID1  | SID0   | —     | EXIDEN | —     | EID17  | EID16  | <a href="#">631</a> |
| 37B0h | RXF4SIDH    | SID10       | SID9  | SID8   | SID7  | SID6   | SID5  | SID4   | SID3   | <a href="#">631</a> |
| 37AFh | RXF3EIDL    | EID7        | EID6  | EID5   | EID4  | EID3   | EID2  | EID1   | EID0   | <a href="#">632</a> |
| 37AEh | RXF3EIDH    | EID15       | EID14 | EID13  | EID12 | EID11  | EID10 | EID9   | EID8   | <a href="#">632</a> |
| 37ADh | RXF3SIDL    | SID2        | SID1  | SID0   | —     | EXIDEN | —     | EID17  | EID16  | <a href="#">631</a> |
| 37ACh | RXF3SIDH    | SID10       | SID9  | SID8   | SID7  | SID6   | SID5  | SID4   | SID3   | <a href="#">631</a> |
| 37ABh | RXF2EIDL    | EID7        | EID6  | EID5   | EID4  | EID3   | EID2  | EID1   | EID0   | <a href="#">632</a> |
| 37AAh | RXF2EIDH    | EID15       | EID14 | EID13  | EID12 | EID11  | EID10 | EID9   | EID8   | <a href="#">632</a> |
| 37A9h | RXF2SIDL    | SID2        | SID1  | SID0   | —     | EXIDEN | —     | EID17  | EID16  | <a href="#">631</a> |
| 37A8h | RXF2SIDH    | SID10       | SID9  | SID8   | SID7  | SID6   | SID5  | SID4   | SID3   | <a href="#">631</a> |
| 37A7h | RXF1EIDL    | EID7        | EID6  | EID5   | EID4  | EID3   | EID2  | EID1   | EID0   | <a href="#">632</a> |
| 37A6h | RXF1EIDH    | EID15       | EID14 | EID13  | EID12 | EID11  | EID10 | EID9   | EID8   | <a href="#">632</a> |
| 37A5h | RXF1SIDL    | SID2        | SID1  | SID0   | —     | EXIDEN | —     | EID17  | EID16  | <a href="#">631</a> |
| 37A4h | RXF1SIDH    | SID10       | SID9  | SID8   | SID7  | SID6   | SID5  | SID4   | SID3   | <a href="#">631</a> |
| 37A3h | RXF0EIDL    | EID7        | EID6  | EID5   | EID4  | EID3   | EID2  | EID1   | EID0   | <a href="#">632</a> |
| 37A2h | RXF0EIDH    | EID15       | EID14 | EID13  | EID12 | EID11  | EID10 | EID9   | EID8   | <a href="#">632</a> |
| 37A1h | RXF0SIDL    | SID2        | SID1  | SID0   | —     | EXIDEN | —     | EID17  | EID16  | <a href="#">631</a> |

**Legend:** x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

**Note 1:** Not present in LF devices.

# PIC18(L)F25/26K83

**TABLE 43-1: REGISTER FILE SUMMARY FOR PIC18(L)F25/26K83 DEVICES (CONTINUED)**

| Addr  | Name        | Bit 7       | Bit 6 | Bit 5   | Bit 4   | Bit 3   | Bit 2   | Bit 1   | Bit 0   | Register on page    |
|-------|-------------|-------------|-------|---------|---------|---------|---------|---------|---------|---------------------|
| 37A0h | RXF0SIDH    | SID10       | SID9  | SID8    | SID7    | SID6    | SID5    | SID4    | SID3    | <a href="#">631</a> |
| 379Fh | CANCON_RO4  | CANCON_RO4  |       |         |         |         |         |         |         | <a href="#">604</a> |
| 379Eh | CANSTAT_RO4 | CANSTAT_RO4 |       |         |         |         |         |         |         | <a href="#">605</a> |
| 379Dh | B5D7        | B5D7        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 379Ch | B5D6        | B5D6        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 379Bh | B5D5        | B5D5        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 379Ah | B5D4        | B5D4        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 3799h | B5D3        | B5D3        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 3798h | B5D2        | B5D2        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 3797h | B5D1        | B5D1        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 3796h | B5D0        | B5D0        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 3795h | B5DLC       | —           | RXRTR | RB1     | RB0     | DLC3    | DLC2    | DLC1    | DLC0    | <a href="#">629</a> |
| 3795h | B5DLC       | —           | TXRTR | —       | —       | DLC3    | DLC2    | DLC1    | DLC0    | <a href="#">630</a> |
| 3794h | B5EIDL      | EID7        | EID6  | EID5    | EID4    | EID3    | EID2    | EID1    | EID0    | <a href="#">627</a> |
| 3793h | B5EIDH      | EID15       | EID14 | EID13   | EID12   | EID11   | EID10   | EID9    | EID8    | <a href="#">627</a> |
| 3792h | B5SIDL      | SID2        | SID1  | SID0    | SRR     | EXIDE   | —       | EID17   | EID16   | <a href="#">626</a> |
| 3791h | B5SIDH      | SID10       | SID9  | SID8    | SID7    | SID6    | SID5    | SID4    | SID3    | <a href="#">625</a> |
| 3790h | B5CON       | RXFUL       | RXM1  | RXRTRRO | FILHIT4 | FILHIT3 | FILHIT2 | FILHIT1 | FILHIT0 | <a href="#">623</a> |
| 3790h | B5CON       | TXBIF       | TXABT | TXLARB  | TXERR   | TXREQ   | RTREN   | TXPRI1  | TXPRI0  | <a href="#">624</a> |
| 378Fh | CANCON_RO5  | CANCON_RO5  |       |         |         |         |         |         |         | <a href="#">604</a> |
| 378Eh | CANSTAT_RO5 | CANSTAT_RO5 |       |         |         |         |         |         |         | <a href="#">605</a> |
| 378Dh | B4D7        | B4D7        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 378Ch | B4D6        | B4D6        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 378Bh | B4D5        | B4D5        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 378Ah | B4D4        | B4D4        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 3789h | B4D3        | B4D3        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 3788h | B4D2        | B4D2        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 3787h | B4D1        | B4D1        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 3786h | B4D0        | B4D0        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 3785h | B4DLC       | —           | RXRTR | RB1     | RB0     | DLC3    | DLC2    | DLC1    | DLC0    | <a href="#">629</a> |
| 3785h | B4DLC       | —           | TXRTR | —       | —       | DLC3    | DLC2    | DLC1    | DLC0    | <a href="#">630</a> |
| 3784h | B4EIDL      | EID7        | EID6  | EID5    | EID4    | EID3    | EID2    | EID1    | EID0    | <a href="#">627</a> |
| 3783h | B4EIDH      | EID15       | EID14 | EID13   | EID12   | EID11   | EID10   | EID9    | EID8    | <a href="#">627</a> |
| 3782h | B4SIDL      | SID2        | SID1  | SID0    | SRR     | EXIDE   | —       | EID17   | EID16   | <a href="#">626</a> |
| 3781h | B4SIDH      | SID10       | SID9  | SID8    | SID7    | SID6    | SID5    | SID4    | SID3    | <a href="#">625</a> |
| 3780h | B4CON       | RXFUL       | RXM1  | RXRTRRO | FILHIT4 | FILHIT3 | FILHIT2 | FILHIT1 | FILHIT0 | <a href="#">623</a> |
| 3780h | B4CON       | TXBIF       | TXABT | TXLARB  | TXERR   | TXREQ   | RTREN   | TXPRI1  | TXPRI0  | <a href="#">624</a> |
| 377Fh | CANCON_RO6  | CANCON_RO6  |       |         |         |         |         |         |         | <a href="#">604</a> |
| 377Eh | CANSTAT_RO6 | CANSTAT_RO6 |       |         |         |         |         |         |         | <a href="#">605</a> |
| 377Dh | B3D7        | B3D7        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 377Ch | B3D6        | B3D6        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 377Bh | B3D5        | B3D5        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 377Ah | B3D4        | B3D4        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 3779h | B3D3        | B3D3        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 3778h | B3D2        | B3D2        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 3777h | B3D1        | B3D1        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 3776h | B3D0        | B3D0        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 3775h | B3DLC       | —           | RXRTR | RB1     | RB0     | DLC3    | DLC2    | DLC1    | DLC0    | <a href="#">629</a> |
| 3775h | B3DLC       | —           | TXRTR | —       | —       | DLC3    | DLC2    | DLC1    | DLC0    | <a href="#">630</a> |
| 3774h | B3EIDL      | EID7        | EID6  | EID5    | EID4    | EID3    | EID2    | EID1    | EID0    | <a href="#">627</a> |

**Legend:** x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

**Note 1:** Not present in LF devices.

# PIC18(L)F25/26K83

**TABLE 43-1: REGISTER FILE SUMMARY FOR PIC18(L)F25/26K83 DEVICES (CONTINUED)**

| Addr  | Name        | Bit 7       | Bit 6 | Bit 5   | Bit 4   | Bit 3   | Bit 2   | Bit 1   | Bit 0   | Register on page    |
|-------|-------------|-------------|-------|---------|---------|---------|---------|---------|---------|---------------------|
| 3773h | B3EIDH      | EID15       | EID14 | EID13   | EID12   | EID11   | EID10   | EID9    | EID8    | <a href="#">627</a> |
| 3772h | B3SIDL      | SID2        | SID1  | SID0    | SRR     | EXIDE   | —       | EID17   | EID16   | <a href="#">626</a> |
| 3771h | B3SIDH      | SID10       | SID9  | SID8    | SID7    | SID6    | SID5    | SID4    | SID3    | <a href="#">625</a> |
| 3770h | B3CON       | RXFUL       | RXM1  | RXRTRRO | FILHIT4 | FILHIT3 | FILHIT2 | FILHIT1 | FILHIT0 | <a href="#">623</a> |
| 3770h | B3CON       | TXBIF       | TXABT | TXLARB  | TXERR   | TXREQ   | RTREN   | TXPRI1  | TXPRI0  | <a href="#">624</a> |
| 376Fh | CANCON_RO7  | CANCON_RO7  |       |         |         |         |         |         |         | <a href="#">604</a> |
| 376Eh | CANSTAT_RO7 | CANSTAT_RO7 |       |         |         |         |         |         |         | <a href="#">605</a> |
| 376Dh | B2D7        | B2D7        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 376Ch | B2D6        | B2D6        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 376Bh | B2D5        | B2D5        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 376Ah | B2D4        | B2D4        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 3769h | B2D3        | B2D3        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 3768h | B2D2        | B2D2        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 3767h | B2D1        | B2D1        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 3766h | B2D0        | B2D0        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 3765h | B2DLC       | —           | RXRTR | RB1     | RB0     | DLC3    | DLC2    | DLC1    | DLC0    | <a href="#">629</a> |
| 3765h | B2DLC       | —           | TXRTR | —       | —       | DLC3    | DLC2    | DLC1    | DLC0    | <a href="#">630</a> |
| 3764h | B2EIDL      | EID7        | EID6  | EID5    | EID4    | EID3    | EID2    | EID1    | EID0    | <a href="#">627</a> |
| 3763h | B2EIDH      | EID15       | EID14 | EID13   | EID12   | EID11   | EID10   | EID9    | EID8    | <a href="#">627</a> |
| 3762h | B2SIDL      | SID2        | SID1  | SID0    | SRR     | EXIDE   | —       | EID17   | EID16   | <a href="#">626</a> |
| 3761h | B2SIDH      | SID10       | SID9  | SID8    | SID7    | SID6    | SID5    | SID4    | SID3    | <a href="#">625</a> |
| 3760h | B2CON       | RXFUL       | RXM1  | RXRTRRO | FILHIT4 | FILHIT3 | FILHIT2 | FILHIT1 | FILHIT0 | <a href="#">623</a> |
| 3760h | B2CON       | TXBIF       | TXABT | TXLARB  | TXERR   | TXREQ   | RTREN   | TXPRI1  | TXPRI0  | <a href="#">624</a> |
| 375Fh | CANCON_RO8  | CANCON_RO8  |       |         |         |         |         |         |         | <a href="#">604</a> |
| 375Eh | CANSTAT_RO8 | CANSTAT_RO8 |       |         |         |         |         |         |         | <a href="#">605</a> |
| 375Dh | B1D7        | B1D7        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 375Ch | B1D6        | B1D6        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 375Bh | B1D5        | B1D5        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 375Ah | B1D4        | B1D4        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 3759h | B1D3        | B1D3        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 3758h | B1D2        | B1D2        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 3757h | B1D1        | B1D1        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 3756h | B1D0        | B1D0        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 3755h | B1DLC       | —           | RXRTR | RB1     | RB0     | DLC3    | DLC2    | DLC1    | DLC0    | <a href="#">629</a> |
| 3755h | B1DLC       | —           | TXRTR | —       | —       | DLC3    | DLC2    | DLC1    | DLC0    | <a href="#">630</a> |
| 3754h | B1EIDL      | EID7        | EID6  | EID5    | EID4    | EID3    | EID2    | EID1    | EID0    | <a href="#">627</a> |
| 3753h | B1EIDH      | EID15       | EID14 | EID13   | EID12   | EID11   | EID10   | EID9    | EID8    | <a href="#">627</a> |
| 3752h | B1SIDL      | SID2        | SID1  | SID0    | SRR     | EXIDE   | —       | EID17   | EID16   | <a href="#">626</a> |
| 3751h | B1SIDH      | SID10       | SID9  | SID8    | SID7    | SID6    | SID5    | SID4    | SID3    | <a href="#">625</a> |
| 3750h | B1CON       | RXFUL       | RXM1  | RXRTRRO | FILHIT4 | FILHIT3 | FILHIT2 | FILHIT1 | FILHIT0 | <a href="#">623</a> |
| 3750h | B1CON       | TXBIF       | TXABT | TXLARB  | TXERR   | TXREQ   | RTREN   | TXPRI1  | TXPRI0  | <a href="#">624</a> |
| 374Fh | CANCON_RO9  | CANCON_RO9  |       |         |         |         |         |         |         | <a href="#">604</a> |
| 374Eh | CANSTAT_RO9 | CANSTAT_RO9 |       |         |         |         |         |         |         | <a href="#">605</a> |
| 374Dh | B0D7        | B0D7        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 374Ch | B0D6        | B0D6        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 374Bh | B0D5        | B0D5        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 374Ah | B0D4        | B0D4        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 3749h | B0D3        | B0D3        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 3748h | B0D2        | B0D2        |       |         |         |         |         |         |         | <a href="#">628</a> |
| 3747h | B0D1        | B0D1        |       |         |         |         |         |         |         | <a href="#">628</a> |

**Legend:** x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

**Note 1:** Not present in LF devices.

# PIC18(L)F25/26K83

**TABLE 43-1: REGISTER FILE SUMMARY FOR PIC18(L)F25/26K83 DEVICES (CONTINUED)**

| Addr  | Name      | Bit 7   | Bit 6   | Bit 5   | Bit 4   | Bit 3   | Bit 2   | Bit 1   | Bit 0   | Register on page    |
|-------|-----------|---------|---------|---------|---------|---------|---------|---------|---------|---------------------|
| 3746h | B0D0      | B0D0    |         |         |         |         |         |         |         | <a href="#">628</a> |
| 3745h | B0DLC     | —       | RXRTR   | RB1     | RB0     | DLC3    | DLC2    | DLC1    | DLC0    | <a href="#">629</a> |
| 3745h | B0DLC     | —       | TXRTR   | —       | —       | DLC3    | DLC2    | DLC1    | DLC0    | <a href="#">630</a> |
| 3744h | B0EIDL    | EID7    | EID6    | EID5    | EID4    | EID3    | EID2    | EID1    | EID0    | <a href="#">627</a> |
| 3743h | B0EIDH    | EID15   | EID14   | EID13   | EID12   | EID11   | EID10   | EID9    | EID8    | <a href="#">627</a> |
| 3742h | B0SIDL    | SID2    | SID1    | SID0    | SRR     | EXIDE   | —       | EID17   | EID16   | <a href="#">626</a> |
| 3741h | B0SIDH    | SID10   | SID9    | SID8    | SID7    | SID6    | SID5    | SID4    | SID3    | <a href="#">625</a> |
| 3740h | B0CON     | RXFUL   | RXM1    | RXRTRRO | FILHIT4 | FILHIT3 | FILHIT2 | FILHIT1 | FILHIT0 | <a href="#">623</a> |
| 3740h | B0CON     | TXBIF   | TXABT   | TXLARB  | TXERR   | TXREQ   | RTREN   | TXPRI1  | TXPRI0  | <a href="#">624</a> |
| 373Fh | TXBIE     | —       | —       | —       | TXB2IE  | TXB1IE  | TXB0IE  | —       | —       | <a href="#">648</a> |
| 373Eh | BIE0      | B5IE    | B4IE    | B3IE    | B2IE    | B1IE    | B0IE    | RXB1IE  | RXB0IE  | <a href="#">649</a> |
| 373Dh | BSEL0     | B5TXEN  | B4TXEN  | B3TXEN  | B2TXEN  | B1TXEN  | B0TXEN  | —       | —       | <a href="#">630</a> |
| 373Ch | MSEL3     | FIL15_1 | FIL15_0 | FIL14_1 | FIL14_0 | FIL13_1 | FIL13_0 | FIL12_1 | FIL12_0 | <a href="#">640</a> |
| 373Bh | MSEL2     | FIL11_1 | FIL11_0 | FIL10_1 | FIL10_0 | FIL9_1  | FIL9_0  | FIL8_1  | FIL8_0  | <a href="#">639</a> |
| 373Ah | MSEL1     | FIL7_1  | FIL7_0  | FIL6_1  | FIL6_0  | FIL5_1  | FIL5_0  | FIL4_1  | FIL4_0  | <a href="#">638</a> |
| 3739h | MSEL0     | FIL3_1  | FIL3_0  | FIL2_1  | FIL2_0  | FIL1_1  | FIL1_0  | FIL0_1  | FIL0_0  | <a href="#">640</a> |
| 3738h | RXFBCON7  | F15BP_3 | F15BP_2 | F15BP_1 | F15BP_0 | F14BP_3 | F14BP_2 | F14BP_1 | F14BP_0 | <a href="#">636</a> |
| 3737h | RXFBCON6  | F13BP_3 | F13BP_2 | F13BP_1 | F13BP_0 | F12BP_3 | F12BP_2 | F12BP_1 | F12BP_0 | <a href="#">636</a> |
| 3736h | RXFBCON5  | F11BP_3 | F11BP_2 | F11BP_1 | F11BP_0 | F10BP_3 | F10BP_2 | F10BP_1 | F10BP_0 | <a href="#">636</a> |
| 3735h | RXFBCON4  | F9BP_3  | F9BP_2  | F9BP_1  | F9BP_0  | F8BP_3  | F8BP_2  | F8BP_1  | F8BP_0  | <a href="#">636</a> |
| 3734h | RXFBCON3  | F7BP_3  | F7BP_2  | F7BP_1  | F7BP_0  | F6BP_3  | F6BP_2  | F6BP_1  | F6BP_0  | <a href="#">636</a> |
| 3733h | RXFBCON2  | F5BP_3  | F5BP_2  | F5BP_1  | F5BP_0  | F4BP_3  | F4BP_2  | F4BP_1  | F4BP_0  | <a href="#">636</a> |
| 3732h | RXFBCON1  | F3BP_3  | F3BP_2  | F3BP_1  | F3BP_0  | F2BP_3  | F2BP_2  | F2BP_1  | F2BP_0  | <a href="#">636</a> |
| 3731h | RXFBCON0  | F1BP_3  | F1BP_2  | F1BP_1  | F1BP_0  | F0BP_3  | F0BP_2  | F0BP_1  | F0BP_0  | <a href="#">636</a> |
| 3730h | SDFLC     | —       | —       | —       | FLC4    | FLC3    | FLC2    | FLC1    | FLC0    | <a href="#">635</a> |
| 372Fh | RXF15EIDL | EID7    | EID6    | EID5    | EID4    | EID3    | EID2    | EID1    | EID0    | <a href="#">632</a> |
| 372Eh | RXF15EIDH | EID15   | EID14   | EID13   | EID12   | EID11   | EID10   | EID9    | EID8    | <a href="#">632</a> |
| 372Dh | RXF15SIDL | SID2    | SID1    | SID0    | —       | EXIDEN  | —       | EID17   | EID16   | <a href="#">631</a> |
| 372Ch | RXF15SIDH | SID10   | SID9    | SID8    | SID7    | SID6    | SID5    | SID4    | SID3    | <a href="#">631</a> |
| 372Bh | RXF14EIDL | EID7    | EID6    | EID5    | EID4    | EID3    | EID2    | EID1    | EID0    | <a href="#">632</a> |
| 372Ah | RXF14EIDH | EID15   | EID14   | EID13   | EID12   | EID11   | EID10   | EID9    | EID8    | <a href="#">632</a> |
| 3729h | RXF14SIDL | SID2    | SID1    | SID0    | —       | EXIDEN  | —       | EID17   | EID16   | <a href="#">631</a> |
| 3728h | RXF14SIDH | SID10   | SID9    | SID8    | SID7    | SID6    | SID5    | SID4    | SID3    | <a href="#">631</a> |
| 3727h | RXF13EIDL | EID7    | EID6    | EID5    | EID4    | EID3    | EID2    | EID1    | EID0    | <a href="#">632</a> |
| 3726h | RXF13EIDH | EID15   | EID14   | EID13   | EID12   | EID11   | EID10   | EID9    | EID8    | <a href="#">632</a> |
| 3725h | RXF13SIDL | SID2    | SID1    | SID0    | —       | EXIDEN  | —       | EID17   | EID16   | <a href="#">631</a> |
| 3724h | RXF13SIDH | SID10   | SID9    | SID8    | SID7    | SID6    | SID5    | SID4    | SID3    | <a href="#">631</a> |
| 3723h | RXF12EIDL | EID7    | EID6    | EID5    | EID4    | EID3    | EID2    | EID1    | EID0    | <a href="#">632</a> |
| 3722h | RXF12EIDH | EID15   | EID14   | EID13   | EID12   | EID11   | EID10   | EID9    | EID8    | <a href="#">632</a> |
| 3721h | RXF12SIDL | SID2    | SID1    | SID0    | —       | EXIDEN  | —       | EID17   | EID16   | <a href="#">631</a> |
| 3720h | RXF12SIDH | SID10   | SID9    | SID8    | SID7    | SID6    | SID5    | SID4    | SID3    | <a href="#">631</a> |
| 371Fh | RXF11EIDL | EID7    | EID6    | EID5    | EID4    | EID3    | EID2    | EID1    | EID0    | <a href="#">632</a> |
| 371Eh | RXF11EIDH | EID15   | EID14   | EID13   | EID12   | EID11   | EID10   | EID9    | EID8    | <a href="#">632</a> |
| 371Dh | RXF11SIDL | SID2    | SID1    | SID0    | —       | EXIDEN  | —       | EID17   | EID16   | <a href="#">631</a> |
| 371Ch | RXF11SIDH | SID10   | SID9    | SID8    | SID7    | SID6    | SID5    | SID4    | SID3    | <a href="#">631</a> |
| 371Bh | RXF10EIDL | EID7    | EID6    | EID5    | EID4    | EID3    | EID2    | EID1    | EID0    | <a href="#">632</a> |
| 371Ah | RXF10EIDH | EID15   | EID14   | EID13   | EID12   | EID11   | EID10   | EID9    | EID8    | <a href="#">632</a> |
| 3719h | RXF10SIDL | SID2    | SID1    | SID0    | —       | EXIDEN  | —       | EID17   | EID16   | <a href="#">631</a> |
| 3718h | RXF10SIDH | SID10   | SID9    | SID8    | SID7    | SID6    | SID5    | SID4    | SID3    | <a href="#">631</a> |
| 3717h | RXF9EIDL  | EID7    | EID6    | EID5    | EID4    | EID3    | EID2    | EID1    | EID0    | <a href="#">632</a> |

**Legend:** x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

**Note 1:** Not present in LF devices.

# PIC18(L)F25/26K83

**TABLE 43-1: REGISTER FILE SUMMARY FOR PIC18(L)F25/26K83 DEVICES (CONTINUED)**

| Addr  | Name     | Bit 7    | Bit 6   | Bit 5   | Bit 4   | Bit 3   | Bit 2   | Bit 1   | Bit 0   | Register on page    |
|-------|----------|----------|---------|---------|---------|---------|---------|---------|---------|---------------------|
| 3716h | RXF9EIDH | EID15    | EID14   | EID13   | EID12   | EID11   | EID10   | EID9    | EID8    | <a href="#">632</a> |
| 3715h | RXF9SIDL | SID2     | SID1    | SID0    | —       | EXIDEN  | —       | EID17   | EID16   | <a href="#">631</a> |
| 3714h | RXF9SIDH | SID10    | SID9    | SID8    | SID7    | SID6    | SID5    | SID4    | SID3    | <a href="#">631</a> |
| 3713h | RXF8EIDL | EID7     | EID6    | EID5    | EID4    | EID3    | EID2    | EID1    | EID0    | <a href="#">632</a> |
| 3712h | RXF8EIDH | EID15    | EID14   | EID13   | EID12   | EID11   | EID10   | EID9    | EID8    | <a href="#">632</a> |
| 3711h | RXF8SIDL | SID2     | SID1    | SID0    | —       | EXIDEN  | —       | EID17   | EID16   | <a href="#">631</a> |
| 3710h | RXF8SIDH | SID10    | SID9    | SID8    | SID7    | SID6    | SID5    | SID4    | SID3    | <a href="#">631</a> |
| 370Fh | RXF7EIDL | EID7     | EID6    | EID5    | EID4    | EID3    | EID2    | EID1    | EID0    | <a href="#">632</a> |
| 370Eh | RXF7EIDH | EID15    | EID14   | EID13   | EID12   | EID11   | EID10   | EID9    | EID8    | <a href="#">632</a> |
| 370Dh | RXF7SIDL | SID2     | SID1    | SID0    | —       | EXIDEN  | —       | EID17   | EID16   | <a href="#">631</a> |
| 370Ch | RXF7SIDH | SID10    | SID9    | SID8    | SID7    | SID6    | SID5    | SID4    | SID3    | <a href="#">631</a> |
| 370Bh | RXF6EIDL | EID7     | EID6    | EID5    | EID4    | EID3    | EID2    | EID1    | EID0    | <a href="#">632</a> |
| 370Ah | RXF6EIDH | EID15    | EID14   | EID13   | EID12   | EID11   | EID10   | EID9    | EID8    | <a href="#">632</a> |
| 3709h | RXF6SIDL | SID2     | SID1    | SID0    | —       | EXIDEN  | —       | EID17   | EID16   | <a href="#">631</a> |
| 3708h | RXF6SIDH | SID10    | SID9    | SID8    | SID7    | SID6    | SID5    | SID4    | SID3    | <a href="#">631</a> |
| 3707h | RXFCON1  | RXF15EN  | RXF14EN | RXF13EN | RXF12EN | RXF11EN | RXF10EN | RXF9EN  | RXF8EN  | <a href="#">634</a> |
| 3706h | RXFCON0  | RXF7EN   | RXF6EN  | RXF5EN  | RXF4EN  | RXF3EN  | RXF2EN  | RXF1EN  | RXF0EN  | <a href="#">634</a> |
| 3705h | BRGCON3  | WAKDIS   | WAKFIL  | —       | —       | —       | SEG2PH2 | SEG2PH1 | SEG2PH0 | <a href="#">643</a> |
| 3704h | BRGCON2  | SEG2PHTS | SAM     | SEG1PH2 | SEG1PH1 | SEG1PH0 | PRSEG2  | PRSEG1  | PRSEG0  | <a href="#">642</a> |
| 3703h | BRGCON1  | SJW1     | SJW0    | BRP5    | BRP4    | BRP3    | BRP2    | BRP1    | BRP0    | <a href="#">641</a> |
| 3702h | TXERRCNT | TEC7     | TEC6    | TEC5    | TEC4    | TEC3    | TEC2    | TEC1    | TEC0    | <a href="#">613</a> |
| 3701h | RXERRCNT | REC7     | REC6    | REC5    | REC4    | REC3    | REC2    | REC1    | REC0    | <a href="#">622</a> |
| 3700h | CIOCON   | TX1SRC   | —       | —       | —       | —       | —       | —       | CLKSEL  | <a href="#">644</a> |

**Legend:** x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

**Note 1:** Not present in LF devices.

## 44.0 DEVELOPMENT SUPPORT

The PIC® microcontrollers (MCU) and dsPIC® digital signal controllers (DSC) are supported with a full range of software and hardware development tools:

- Integrated Development Environment
  - MPLAB® X IDE Software
- Compilers/Assemblers/Linkers
  - MPLAB XC Compiler
  - MPASM™ Assembler
  - MPLINK™ Object Linker/  
MPLIB™ Object Librarian
  - MPLAB Assembler/Linker/Librarian for  
Various Device Families
- Simulators
  - MPLAB X SIM Software Simulator
- Emulators
  - MPLAB REAL ICE™ In-Circuit Emulator
- In-Circuit Debuggers/Programmers
  - MPLAB ICD 3
  - PICKit™ 3
- Device Programmers
  - MPLAB PM3 Device Programmer
- Low-Cost Demonstration/Development Boards,  
Evaluation Kits and Starter Kits
- Third-party development tools

## 44.1 MPLAB X Integrated Development Environment Software

The MPLAB X IDE is a single, unified graphical user interface for Microchip and third-party software, and hardware development tool that runs on Windows®, Linux and Mac OS® X. Based on the NetBeans IDE, MPLAB X IDE is an entirely new IDE with a host of free software components and plug-ins for high-performance application development and debugging. Moving between tools and upgrading from software simulators to hardware debugging and programming tools is simple with the seamless user interface.

With complete project management, visual call graphs, a configurable watch window and a feature-rich editor that includes code completion and context menus, MPLAB X IDE is flexible and friendly enough for new users. With the ability to support multiple tools on multiple projects with simultaneous debugging, MPLAB X IDE is also suitable for the needs of experienced users.

Feature-Rich Editor:

- Color syntax highlighting
- Smart code completion makes suggestions and provides hints as you type
- Automatic code formatting based on user-defined rules
- Live parsing

User-Friendly, Customizable Interface:

- Fully customizable interface: toolbars, toolbar buttons, windows, window placement, etc.
- Call graph window

Project-Based Workspaces:

- Multiple projects
- Multiple tools
- Multiple configurations
- Simultaneous debugging sessions

File History and Bug Tracking:

- Local file history feature
- Built-in support for Bugzilla issue tracker

## 44.2 MPLAB XC Compilers

The MPLAB XC Compilers are complete ANSI C compilers for all of Microchip's 8, 16, and 32-bit MCU and DSC devices. These compilers provide powerful integration capabilities, superior code optimization and ease of use. MPLAB XC Compilers run on Windows, Linux or MAC OS X.

For easy source level debugging, the compilers provide debug information that is optimized to the MPLAB X IDE.

The free MPLAB XC Compiler editions support all devices and commands, with no time or memory restrictions, and offer sufficient code optimization for most applications.

MPLAB XC Compilers include an assembler, linker and utilities. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. MPLAB XC Compiler uses the assembler to produce its object file. Notable features of the assembler include:

- Support for the entire device instruction set
- Support for fixed-point and floating-point data
- Command-line interface
- Rich directive set
- Flexible macro language
- MPLAB X IDE compatibility

## 44.3 MPASM Assembler

The MPASM Assembler is a full-featured, universal macro assembler for PIC10/12/16/18 MCUs.

The MPASM Assembler generates relocatable object files for the MPLINK Object Linker, Intel® standard HEX files, MAP files to detail memory usage and symbol reference, absolute LST files that contain source lines and generated machine code, and COFF files for debugging.

The MPASM Assembler features include:

- Integration into MPLAB X IDE projects
- User-defined macros to streamline assembly code
- Conditional assembly for multipurpose source files
- Directives that allow complete control over the assembly process

## 44.4 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK Object Linker combines relocatable objects created by the MPASM Assembler. It can link relocatable objects from precompiled libraries, using directives from a linker script.

The MPLIB Object Librarian manages the creation and modification of library files of precompiled code. When a routine from a library is called from a source file, only the modules that contain that routine will be linked in with the application. This allows large libraries to be used efficiently in many different applications.

The object linker/librarian features include:

- Efficient linking of single libraries instead of many smaller files
- Enhanced code maintainability by grouping related modules together
- Flexible creation of libraries with easy module listing, replacement, deletion and extraction

## 44.5 MPLAB Assembler, Linker and Librarian for Various Device Families

MPLAB Assembler produces relocatable machine code from symbolic assembly language for PIC24, PIC32 and dsPIC DSC devices. MPLAB XC Compiler uses the assembler to produce its object file. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. Notable features of the assembler include:

- Support for the entire device instruction set
- Support for fixed-point and floating-point data
- Command-line interface
- Rich directive set
- Flexible macro language
- MPLAB X IDE compatibility

## 44.6 MPLAB X SIM Software Simulator

The MPLAB X SIM Software Simulator allows code development in a PC-hosted environment by simulating the PIC MCUs and dsPIC DSCs on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a comprehensive stimulus controller. Registers can be logged to files for further run-time analysis. The trace buffer and logic analyzer display extend the power of the simulator to record and track program execution, actions on I/O, most peripherals and internal registers.

The MPLAB X SIM Software Simulator fully supports symbolic debugging using the MPLAB XC Compilers, and the MPASM and MPLAB Assemblers. The software simulator offers the flexibility to develop and debug code outside of the hardware laboratory environment, making it an excellent, economical software development tool.

## 44.7 MPLAB REAL ICE In-Circuit Emulator System

The MPLAB REAL ICE In-Circuit Emulator System is Microchip's next generation high-speed emulator for Microchip Flash DSC and MCU devices. It debugs and programs all 8, 16 and 32-bit MCU, and DSC devices with the easy-to-use, powerful graphical user interface of the MPLAB X IDE.

The emulator is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with either a connector compatible with in-circuit debugger systems (RJ-11) or with the new high-speed, noise tolerant, Low-Voltage Differential Signal (LVDS) interconnection (CAT5).

The emulator is field upgradable through future firmware downloads in MPLAB X IDE. MPLAB REAL ICE offers significant advantages over competitive emulators including full-speed emulation, run-time variable watches, trace analysis, complex breakpoints, logic probes, a ruggedized probe interface and long (up to three meters) interconnection cables.

## 44.8 MPLAB ICD 3 In-Circuit Debugger System

The MPLAB ICD 3 In-Circuit Debugger System is Microchip's most cost-effective, high-speed hardware debugger/programmer for Microchip Flash DSC and MCU devices. It debugs and programs PIC Flash microcontrollers and dsPIC DSCs with the powerful, yet easy-to-use graphical user interface of the MPLAB IDE.

The MPLAB ICD 3 In-Circuit Debugger probe is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with a connector compatible with the MPLAB ICD 2 or MPLAB REAL ICE systems (RJ-11). MPLAB ICD 3 supports all MPLAB ICD 2 headers.

## 44.9 PICkit 3 In-Circuit Debugger/Programmer

The MPLAB PICkit 3 allows debugging and programming of PIC and dsPIC Flash microcontrollers at a most affordable price point using the powerful graphical user interface of the MPLAB IDE. The MPLAB PICkit 3 is connected to the design engineer's PC using a full-speed USB interface and can be connected to the target via a Microchip debug (RJ-11) connector (compatible with MPLAB ICD 3 and MPLAB REAL ICE). The connector uses two device I/O pins and the Reset line to implement in-circuit debugging and In-Circuit Serial Programming™ (ICSP™).

## 44.10 MPLAB PM3 Device Programmer

The MPLAB PM3 Device Programmer is a universal, CE compliant device programmer with programmable voltage verification at VDDMIN and VDDMAX for maximum reliability. It features a large LCD display (128 x 64) for menus and error messages, and a modular, detachable socket assembly to support various package types. The ICSP cable assembly is included as a standard item. In Stand-Alone mode, the MPLAB PM3 Device Programmer can read, verify and program PIC devices without a PC connection. It can also set code protection in this mode. The MPLAB PM3 connects to the host PC via an RS-232 or USB cable. The MPLAB PM3 has high-speed communications and optimized algorithms for quick programming of large memory devices, and incorporates an MMC card for file storage and data applications.



## 44.11 Demonstration/Development Boards, Evaluation Kits, and Starter Kits

A wide variety of demonstration, development and evaluation boards for various PIC MCUs and dsPIC DSCs allows quick application development on fully functional systems. Most boards include prototyping areas for adding custom circuitry and provide application firmware and source code for examination and modification.

The boards support a variety of features, including LEDs, temperature sensors, switches, speakers, RS-232 interfaces, LCD displays, potentiometers and additional EEPROM memory.

The demonstration and development boards can be used in teaching environments, for prototyping custom circuits and for learning about various microcontroller applications.

In addition to the PICDEM™ and dsPICDEM™ demonstration/development board series of circuits, Microchip has a line of evaluation kits and demonstration software for analog filter design, KEELOQ® security ICs, CAN, IrDA®, PowerSmart battery management, SEEVAL® evaluation system, Sigma-Delta ADC, flow rate sensing, plus many more.

Also available are starter kits that contain everything needed to experience the specified device. This usually includes a single application and debug capability, all on one board.

Check the Microchip web page ([www.microchip.com](http://www.microchip.com)) for the complete list of demonstration, development and evaluation kits.

## 44.12 Third-Party Development Tools

Microchip also offers a great collection of tools from third-party vendors. These tools are carefully selected to offer good value and unique functionality.

- Device Programmers and Gang Programmers from companies, such as SoftLog and CCS
- Software Tools from companies, such as Gimpel and Trace Systems
- Protocol Analyzers from companies, such as Saleae and Total Phase
- Demonstration Boards from companies, such as MikroElektronika, Digilent® and Olimex
- Embedded Ethernet Solutions from companies, such as EZ Web Lynx, WIZnet and IPLogika®

## 45.0 ELECTRICAL SPECIFICATIONS

### 45.1 Absolute Maximum Ratings<sup>(†)</sup>

|                                                                                                   |                                   |
|---------------------------------------------------------------------------------------------------|-----------------------------------|
| Ambient temperature under bias .....                                                              | -40°C to +125°C                   |
| Storage temperature .....                                                                         | -65°C to +150°C                   |
| Voltage on pins with respect to V <sub>SS</sub>                                                   |                                   |
| on V <sub>DD</sub> pin                                                                            |                                   |
| PIC18F25/26K83 .....                                                                              | -0.3V to +6.5V                    |
| PIC18LF25/26K83 .....                                                                             | -0.3V to +4.0V                    |
| on MCLR pin .....                                                                                 | -0.3V to +9.0V                    |
| on all other pins .....                                                                           | -0.3V to (V <sub>DD</sub> + 0.3V) |
| Maximum current                                                                                   |                                   |
| on V <sub>SS</sub> pin <sup>(1)</sup>                                                             |                                   |
| -40°C ≤ T <sub>A</sub> ≤ +85°C .....                                                              | 350 mA                            |
| 85°C < T <sub>A</sub> ≤ +125°C .....                                                              | 120 mA                            |
| on V <sub>DD</sub> pin for 28-Pin devices <sup>(1)</sup>                                          |                                   |
| -40°C ≤ T <sub>A</sub> ≤ +85°C .....                                                              | 250 mA                            |
| 85°C < T <sub>A</sub> ≤ +125°C .....                                                              | 85 mA                             |
| on any standard I/O pin .....                                                                     | ±50 mA                            |
| Clamp current, I <sub>K</sub> (V <sub>PIN</sub> < 0 or V <sub>PIN</sub> > V <sub>DD</sub> ) ..... | ±20 mA                            |
| Total power dissipation <sup>(2)</sup> .....                                                      | 800 mW                            |

**Note 1:** Maximum current rating requires even load distribution across I/O pins. Maximum current rating may be limited by the device package power dissipation characterizations, see [Table 45-6](#) to calculate device specifications.

**2:** Power dissipation is calculated as follows:

$$P_{DIS} = V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD} - V_{OH}) \times I_{OH}\} + \sum (V_{OI} \times I_{OL})$$

† NOTICE: Stresses above those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure above maximum rating conditions for extended periods may affect device reliability.

## 45.2 Standard Operating Conditions

The standard operating conditions for any device are defined as:

Operating Voltage:  $V_{DDMIN} \leq V_{DD} \leq V_{DDMAX}$

Operating Temperature:  $T_{A\_MIN} \leq T_A \leq T_{A\_MAX}$

### V<sub>DD</sub> — Operating Supply Voltage<sup>(1)</sup>

#### PIC18LF25/26K83

|                                                      |       |
|------------------------------------------------------|-------|
| V <sub>DDMIN</sub> (F <sub>osc</sub> ≤ 16 MHz) ..... | +1.8V |
| V <sub>DDMIN</sub> (F <sub>osc</sub> ≤ 32 MHz) ..... | +2.5V |
| V <sub>DDMIN</sub> (F <sub>osc</sub> ≤ 64 MHz) ..... | +2.7V |
| V <sub>DDMAX</sub> .....                             | +3.6V |

#### PIC18F25/26K83

|                                                      |       |
|------------------------------------------------------|-------|
| V <sub>DDMIN</sub> (F <sub>osc</sub> ≤ 16 MHz) ..... | +2.3V |
| V <sub>DDMIN</sub> (F <sub>osc</sub> ≤ 32 MHz) ..... | +2.5V |
| V <sub>DDMIN</sub> (F <sub>osc</sub> ≤ 64 MHz) ..... | +3.0V |
| V <sub>DDMAX</sub> .....                             | +5.5V |

### T<sub>A</sub> — Operating Ambient Temperature Range

#### Industrial Temperature

|                           |       |
|---------------------------|-------|
| T <sub>A\_MIN</sub> ..... | -40°C |
| T <sub>A\_MAX</sub> ..... | +85°C |

#### Extended Temperature

|                           |        |
|---------------------------|--------|
| T <sub>A\_MIN</sub> ..... | -40°C  |
| T <sub>A\_MAX</sub> ..... | +125°C |

**Note 1:** See Parameter [Supply Voltage](#), DS Characteristics: Supply Voltage.

**FIGURE 45-1: VOLTAGE FREQUENCY GRAPH,  $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ , PIC18F25/26K83 ONLY**

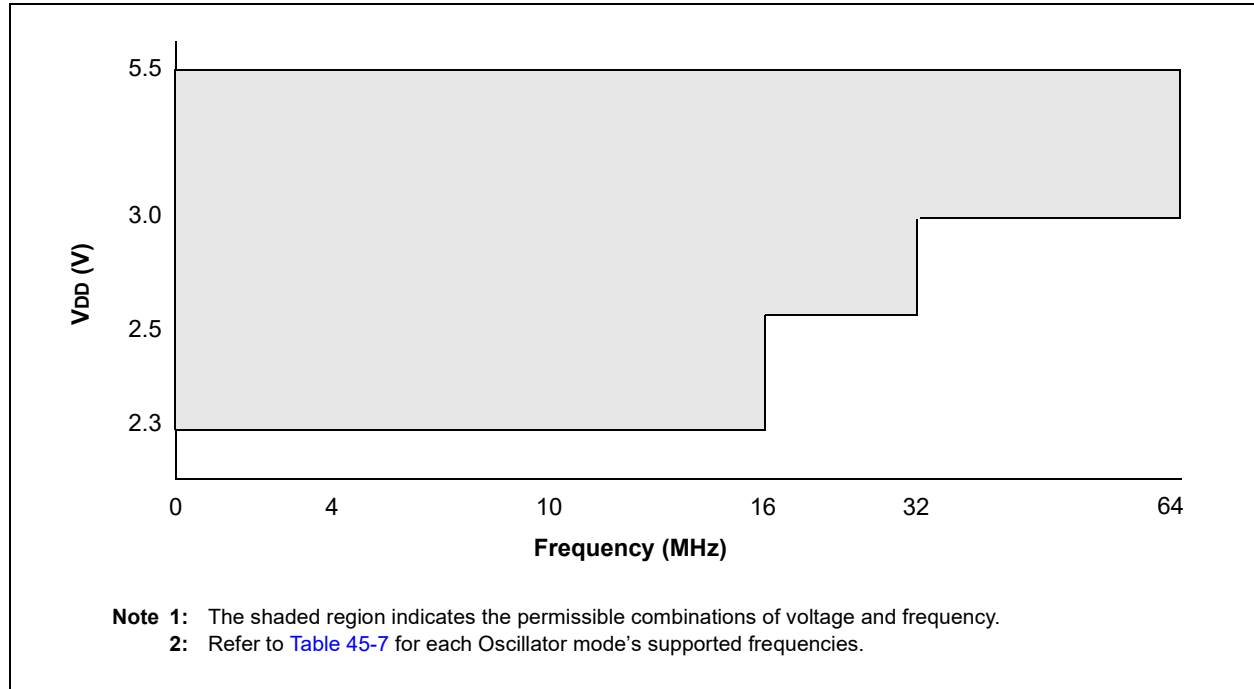
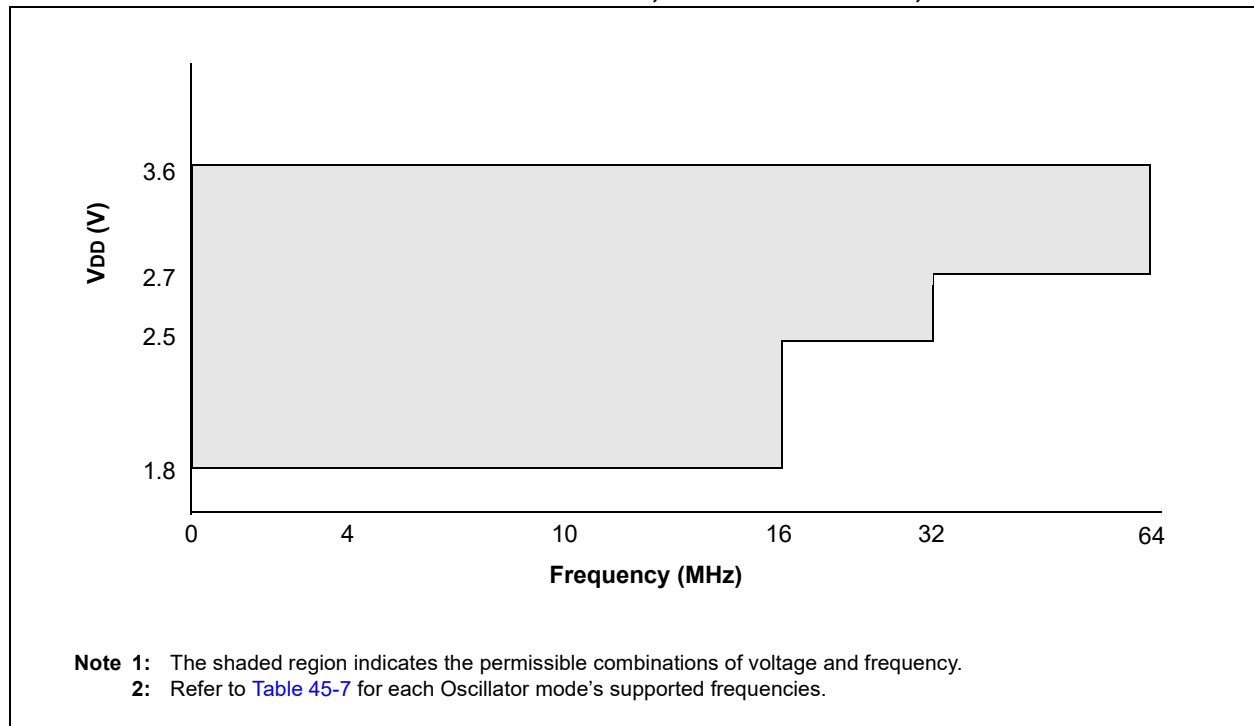


**FIGURE 45-2: VOLTAGE FREQUENCY GRAPH,  $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ , PIC18LF25/26K83 ONLY**



## 45.3 DC Characteristics

TABLE 45-1: SUPPLY VOLTAGE

| PIC18LF25/26K83                                                       |       |                | Standard Operating Conditions (unless otherwise stated) |       |      |       |                                      |
|-----------------------------------------------------------------------|-------|----------------|---------------------------------------------------------|-------|------|-------|--------------------------------------|
| PIC18F25/26K83                                                        |       |                |                                                         |       |      |       |                                      |
| Param. No.                                                            | Sym.  | Characteristic | Min.                                                    | Typ.† | Max. | Units | Conditions                           |
| Supply Voltage                                                        |       |                |                                                         |       |      |       |                                      |
| D002                                                                  | VDD   |                | 2.3                                                     | —     | 3.6  | V     | FOSC ≤ 16 MHz (-40°C <+ 25°C)        |
|                                                                       |       |                | 1.8                                                     | —     | 3.6  | V     | FOSC ≤ 16 MHz (≥25°C to 125°C)       |
|                                                                       |       |                | 2.5                                                     | —     | 3.6  | V     | FOSC > 16 MHz                        |
|                                                                       |       |                | 2.7                                                     | —     | 3.6  | V     | FOSC > 32 MHz                        |
| D002                                                                  | VDD   |                | 2.3                                                     | —     | 5.5  | V     | FOSC ≤ 16 MHz                        |
|                                                                       |       |                | 2.5                                                     | —     | 5.5  | V     | FOSC > 16 MHz                        |
|                                                                       |       |                | 3.0                                                     | —     | 5.5  | V     | FOSC > 32 MHz                        |
| RAM Data Retention <sup>(1)</sup>                                     |       |                |                                                         |       |      |       |                                      |
| D003                                                                  | VDR   |                | 1.5                                                     | —     | —    | V     | Device in Sleep mode                 |
| D003                                                                  | VDR   |                | 1.7                                                     | —     | —    | V     | Device in Sleep mode                 |
| Power-on Reset Release Voltage <sup>(2)</sup>                         |       |                |                                                         |       |      |       |                                      |
| D004                                                                  | VPOR  |                | —                                                       | 1.6   | —    | V     | BOR or LPBOR disabled <sup>(3)</sup> |
| D004                                                                  | VPOR  |                | —                                                       | 1.6   | —    | V     | BOR or LPBOR disabled <sup>(3)</sup> |
| Power-on Reset Rearm Voltage <sup>(2)</sup>                           |       |                |                                                         |       |      |       |                                      |
| D005                                                                  | VPORR |                | —                                                       | 0.8   | —    | V     | BOR or LPBOR disabled <sup>(3)</sup> |
| D005                                                                  | VPORR |                | —                                                       | 1.5   | —    | V     | BOR or LPBOR disabled <sup>(3)</sup> |
| VDD Rise Rate to ensure internal Power-on Reset signal <sup>(2)</sup> |       |                |                                                         |       |      |       |                                      |
| D006                                                                  | SVDD  |                | 0.05                                                    | —     | —    | V/ms  | BOR or LPBOR disabled <sup>(3)</sup> |
| D006                                                                  | SVDD  |                | 0.05                                                    | —     | —    | V/ms  | BOR or LPBOR disabled <sup>(3)</sup> |

† Data in "Typ." column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** This is the limit to which VDD can be lowered in Sleep mode without losing RAM data.

**2:** See [Figure 45-3](#), POR and POR REARM with Slow Rising VDD.

**3:** See [Table 45-11](#) for BOR and LPBOR trip point information.

**FIGURE 45-3: POR AND POR REARM WITH SLOW RISING  $V_{DD}$**

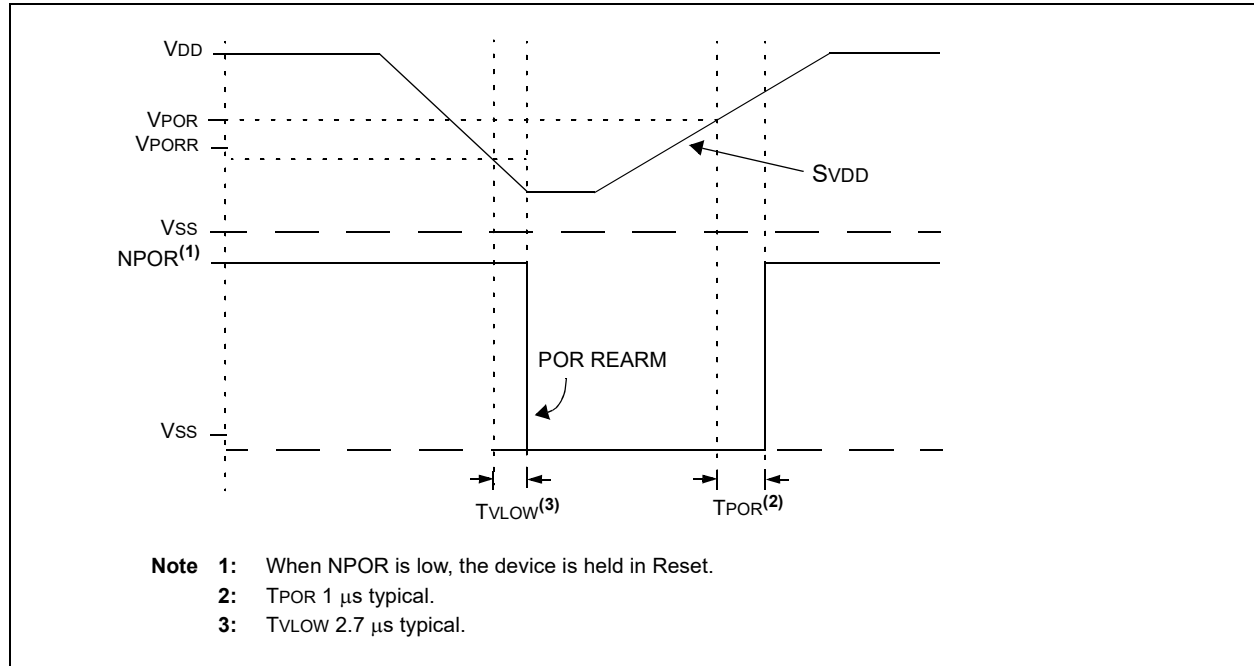


**TABLE 45-2: SUPPLY CURRENT ( $I_{DD}$ )<sup>(1,2,4)</sup>**

| PIC18LF25/26K83 |                       |                                               |      | Standard Operating Conditions (unless otherwise stated) |      |       |                 |               |  |
|-----------------|-----------------------|-----------------------------------------------|------|---------------------------------------------------------|------|-------|-----------------|---------------|--|
| PIC18F25/26K83  |                       |                                               |      |                                                         |      |       |                 |               |  |
| Param. No.      | Symbol                | Device Characteristics                        | Min. | Typ. <sup>†</sup>                                       | Max. | Units | Conditions      |               |  |
|                 |                       |                                               |      |                                                         |      |       | V <sub>DD</sub> | Note          |  |
| D100            | $I_{DD_{XT4}}$        | XT = 4 MHz                                    | —    | 950                                                     | 1200 | μA    | 3.0V            |               |  |
| D100            | $I_{DD_{XT4}}$        | XT = 4 MHz                                    | —    | 1004                                                    | 1300 | μA    | 3.0V            |               |  |
| D100A           | $I_{DD_{XT4}}$        | XT = 4 MHz                                    | —    | 468                                                     | 850  | μA    | 3.0V            | PMD's all 1's |  |
| D100A           | $I_{DD_{XT4}}$        | XT = 4 MHz                                    | —    | 620                                                     | 950  | μA    | 3.0V            | PMD's all 1's |  |
| D101            | $I_{DD_{HFO16}}$      | HFINTOSC = 16 MHz                             | —    | 4.1                                                     | 5.0  | mA    | 3.0V            |               |  |
| D101            | $I_{DD_{HFO16}}$      | HFINTOSC = 16 MHz                             | —    | 4.1                                                     | 5.1  | mA    | 3.0V            |               |  |
| D101A           | $I_{DD_{HFO16}}$      | HFINTOSC = 16 MHz                             | —    | 2.4                                                     | 3.2  | mA    | 3.0V            | PMD's all 1's |  |
| D101A           | $I_{DD_{HFO16}}$      | HFINTOSC = 16 MHz                             | —    | 2.4                                                     | 3.5  | mA    | 3.0V            | PMD's all 1's |  |
| D102            | $I_{DD_{HFOPLL}}$     | HFINTOSC = 64 MHz                             | —    | 15.6                                                    | 18.5 | mA    | 3.0V            |               |  |
| D102            | $I_{DD_{HFOPLL}}$     | HFINTOSC = 64 MHz                             | —    | 16.7                                                    | 19   | mA    | 3.0V            |               |  |
| D102A           | $I_{DD_{HFOPLL}}$     | HFINTOSC = 64 MHz                             | —    | 9                                                       | 11   | mA    | 3.0V            | PMD's all 1's |  |
| D102A           | $I_{DD_{HFOPLL}}$     | HFINTOSC = 64 MHz                             | —    | 9                                                       | 11.5 | mA    | 3.0V            | PMD's all 1's |  |
| D103            | $I_{DD_{HSPLL64}}$    | HS+PLL = 64 MHz                               | —    | 14.8                                                    | 19   | mA    | 3.0V            |               |  |
| D103            | $I_{DD_{HSPLL64}}$    | HS+PLL = 64 MHz                               | —    | 14.8                                                    | 20   | mA    | 3.0V            |               |  |
| D103A           | $I_{DD_{HSPLL64}}$    | HS+PLL = 64 MHz                               | —    | 7.3                                                     | 10   | mA    | 3.0V            | PMD's all 1's |  |
| D103A           | $I_{DD_{HSPLL64}}$    | HS+PLL = 64 MHz                               | —    | 7.3                                                     | 10   | mA    | 3.0V            | PMD's all 1's |  |
| D104            | $I_{DD_{IDLE}}$       | IDLE mode, HFINTOSC = 16 MHz                  | —    | 2.8                                                     | 4.5  | mA    | 3.0V            |               |  |
| D104            | $I_{DD_{IDLE}}$       | IDLE mode, HFINTOSC = 16 MHz                  | —    | 2.9                                                     | 4.5  | mA    | 3.0V            |               |  |
| D105            | $I_{DD_{DOZE}}^{(3)}$ | DOZE mode, HFINTOSC = 16 MHz, Doze Ratio = 16 | —    | 2.9                                                     | —    | mA    | 3.0V            |               |  |
| D105            | $I_{DD_{DOZE}}^{(3)}$ | DOZE mode, HFINTOSC = 16 MHz, Doze Ratio = 16 | —    | 3                                                       | —    | mA    | 3.0V            |               |  |

<sup>†</sup> Data in "Typ." column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

- Note**
- 1: The test conditions for all  $I_{DD}$  measurements in active operation mode are: OSC1 = external square wave, from rail-to-rail; all I/O pins are outputs driven low; MCLR = V<sub>DD</sub>; WDT disabled.
  - 2: The supply current is mainly a function of the operating voltage and frequency. Other factors, such as I/O pin loading and switching rate, oscillator type, internal code execution pattern and temperature, also have an impact on the current consumption.
  - 3:  $I_{DD_{DOZE}} = [I_{DD_{IDLE}} \cdot (N-1)/N] + I_{DD_{HFO16}}/N$  where N = DOZE Ratio ([Register 10-2](#)).
  - 4: PMD bits are all in the default state, no modules are disabled.

**TABLE 45-3: POWER-DOWN CURRENT (IPD)<sup>(1,2)</sup>**

| PIC18LF25/26K83 |           |                                           |      | Standard Operating Conditions (unless otherwise stated)               |            |             |       |            |                                  |
|-----------------|-----------|-------------------------------------------|------|-----------------------------------------------------------------------|------------|-------------|-------|------------|----------------------------------|
| PIC18F25/26K83  |           |                                           |      | Standard Operating Conditions (unless otherwise stated)<br>VREGPM = 1 |            |             |       |            |                                  |
| Param. No.      | Symbol    | Device Characteristics                    | Min. | Typ.†                                                                 | Max. +85°C | Max. +125°C | Units | Conditions |                                  |
|                 |           |                                           |      |                                                                       |            |             |       | VDD        | Note                             |
| D200            | IPD       | IPD Base                                  | —    | 0.07                                                                  | 2          | 6           | μA    | 3.0V       |                                  |
| D200<br>D200A   | IPD       | IPD Base                                  | —    | 0.4                                                                   | 2.5        | 8           | μA    | 3.0V       |                                  |
|                 |           |                                           | —    | 20                                                                    | 37         | 45          | μA    | 3.0V       | VREGPM = 0                       |
| D201            | IPD_WDT   | Low-Frequency Internal Oscillator/<br>WDT | —    | 0.9                                                                   | 2.9        | 9           | μA    | 3.0V       |                                  |
| D201            | IPD_WDT   | Low-Frequency Internal Oscillator/<br>WDT | —    | 1.1                                                                   | 3.3        | 9           | μA    | 3.0V       |                                  |
| D202            | IPD_SOSC  | Secondary Oscillator (SOSC)               | —    | 0.6                                                                   | 2.8        | 13          | μA    | 3.0V       | LP mode                          |
| D202            | IPD_SOSC  | Secondary Oscillator (SOSC)               | —    | 0.8                                                                   | 3.2        | 15          | μA    | 3.0V       | LP mode                          |
| D203            | IPD_FVR   | FVR                                       | —    | 37                                                                    | 70         | 75          | μA    | 3.0V       | FVRCON = 0x81 or 0x84            |
| D203            | IPD_FVR   | FVR                                       | —    | 30                                                                    | 70         | 76          | μA    | 3.0V       | FVRCON = 0x81 or 0x84            |
| D204            | IPD_BOR   | Brown-out Reset (BOR)                     | —    | 9.4                                                                   | 16         | 20          | μA    | 3.0V       |                                  |
| D204            | IPD_BOR   | Brown-out Reset (BOR)                     | —    | 9.4                                                                   | 17         | 21          | μA    | 3.0V       |                                  |
| D205            | IPD_LPBOR | Low-Power Brown-out Reset (LPBOR)         | —    | 0.2                                                                   | 3          | 9           | μA    | 3.0V       |                                  |
| D205            | IPD_LPBOR | Low-Power Brown-out Reset (LPBOR)         | —    | 0.5                                                                   | 3          | 9           | μA    | 3.0V       |                                  |
| D206            | IPD_HLVD  | High/Low Voltage Detect (HLVD)            | —    | 9.5                                                                   | 16         | 19          | μA    | 3.0V       |                                  |
| D206            | IPD_HLVD  | High/Low Voltage Detect (HLVD)            | —    | 9.7                                                                   | 17         | 20          | μA    | 3.0V       |                                  |
| D207            | IPD_ADCA  | ADC - Active                              | —    | 400                                                                   | —          | —           | μA    | 3.0V       | ADC is converting <sup>(4)</sup> |
| D207            | IPD_ADCA  | ADC - Active                              | —    | 400                                                                   | —          | —           | μA    | 3.0V       | ADC is converting <sup>(4)</sup> |
| D208            | IPD_CMP   | Comparator                                | —    | 33                                                                    | 50         | 55          | μA    | 3.0V       |                                  |
| D208            | IPD_CMP   | Comparator                                | —    | 30                                                                    | 50         | 60          | μA    | 3.0V       |                                  |

† Data in "Typ." column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

- Note** 1: The peripheral current is the sum of the base IDD and the additional current consumed when this peripheral is enabled. The peripheral Δ current can be determined by subtracting the base IDD or IPD current from this limit. Max. values should be used when calculating total current consumption.
- 2: The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode with all I/O pins in high-impedance state and tied to VSS.
- 3: All peripheral currents listed are on a per-peripheral basis if more than one instance of a peripheral is available.
- 4: ADC clock source is FRC.



**TABLE 45-4: I/O PORTS**

**Standard Operating Conditions** (unless otherwise stated)

| Param No. | Sym.             | Characteristic                             | Min.                       | Typ† | Max.                 | Units | Conditions                                                                          |
|-----------|------------------|--------------------------------------------|----------------------------|------|----------------------|-------|-------------------------------------------------------------------------------------|
| D300      | V <sub>IL</sub>  | <b>Input Low Voltage</b>                   |                            |      |                      |       |                                                                                     |
|           |                  | I/O PORT:                                  |                            |      |                      |       |                                                                                     |
|           |                  | with TTL buffer                            | —                          | —    | 0.8                  | V     | 4.5V ≤ V <sub>DD</sub> ≤ 5.5V                                                       |
|           |                  |                                            | —                          | —    | 0.15 V <sub>DD</sub> | V     | 1.8V ≤ V <sub>DD</sub> ≤ 4.5V                                                       |
|           |                  | with Schmitt Trigger buffer                | —                          | —    | 0.2 V <sub>DD</sub>  | V     | 2.0V ≤ V <sub>DD</sub> ≤ 5.5V                                                       |
|           |                  | with I <sup>2</sup> C levels               | —                          | —    | 0.3 V <sub>DD</sub>  | V     |                                                                                     |
|           |                  | with SMBus 2.0                             | —                          | —    | 0.8                  | V     | 2.7V ≤ V <sub>DD</sub> ≤ 5.5V                                                       |
|           |                  | with SMBus 3.0                             | —                          | —    | 0.8                  | V     | 1.8V ≤ V <sub>DD</sub> ≤ 5.5V                                                       |
| D306      |                  | MCLR                                       | —                          | —    | 0.2 V <sub>DD</sub>  | V     |                                                                                     |
| D320      | V <sub>IH</sub>  | <b>Input High Voltage</b>                  |                            |      |                      |       |                                                                                     |
|           |                  | I/O PORT:                                  |                            |      |                      |       |                                                                                     |
|           |                  | with TTL buffer                            | 2.0                        | —    | —                    | V     | 4.5V ≤ V <sub>DD</sub> ≤ 5.5V                                                       |
|           |                  |                                            | 0.25 V <sub>DD</sub> + 0.8 | —    | —                    | V     | 1.8V ≤ V <sub>DD</sub> ≤ 4.5V                                                       |
|           |                  | with Schmitt Trigger buffer                | 0.8 V <sub>DD</sub>        | —    | —                    | V     | 2.0V ≤ V <sub>DD</sub> ≤ 5.5V                                                       |
|           |                  | with I <sup>2</sup> C levels               | 0.7 V <sub>DD</sub>        | —    | —                    | V     |                                                                                     |
|           |                  | with SMBus 2.0                             | 2.1                        | —    | —                    | V     | 2.7V ≤ V <sub>DD</sub> ≤ 5.5V                                                       |
|           |                  | with SMBus 3.0                             | 1.35                       | —    | —                    | V     | 1.8V ≤ V <sub>DD</sub> ≤ 5.5V                                                       |
| D326      |                  | MCLR                                       | 0.7 V <sub>DD</sub>        | —    | —                    | V     |                                                                                     |
| D340      | I <sub>IL</sub>  | <b>Input Leakage Current<sup>(1)</sup></b> |                            |      |                      |       |                                                                                     |
|           |                  | I/O Ports                                  | —                          | ± 5  | ± 125                | nA    | V <sub>SS</sub> ≤ V <sub>PIN</sub> ≤ V <sub>DD</sub> , Pin at high-impedance, 85°C  |
|           |                  |                                            | —                          | ± 5  | ± 1000               | nA    | V <sub>SS</sub> ≤ V <sub>PIN</sub> ≤ V <sub>DD</sub> , Pin at high-impedance, 125°C |
|           |                  | MCLR <sup>(2)</sup>                        | —                          | ± 50 | ± 200                | nA    | V <sub>SS</sub> ≤ V <sub>PIN</sub> ≤ V <sub>DD</sub> , Pin at high-impedance, 85°C  |
| D350      | I <sub>PUR</sub> | <b>Weak Pull-up Current</b>                |                            |      |                      |       |                                                                                     |
|           |                  |                                            | 25                         | 120  | 200                  | μA    | V <sub>DD</sub> = 3.0V, V <sub>PIN</sub> = V <sub>SS</sub>                          |
| D360      | V <sub>OL</sub>  | <b>Output Low Voltage</b>                  |                            |      |                      |       |                                                                                     |
|           |                  | I/O ports                                  | —                          | —    | 0.6                  | V     | I <sub>OL</sub> = 10.0mA, V <sub>DD</sub> = 3.0V                                    |
| D370      | V <sub>OH</sub>  | <b>Output High Voltage</b>                 |                            |      |                      |       |                                                                                     |
|           |                  | I/O ports                                  | V <sub>DD</sub> - 0.7      | —    | —                    | V     | I <sub>OH</sub> = 6.0 mA, V <sub>DD</sub> = 3.0V                                    |
| D380      | C <sub>IO</sub>  | All I/O pins                               | —                          | 5    | 50                   | pF    |                                                                                     |

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** Negative current is defined as current sourced by the pin.

**Note 2:** The leakage current on the MCLR pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.

**TABLE 45-5: MEMORY PROGRAMMING SPECIFICATIONS**

| Standard Operating Conditions (unless otherwise stated) |                    |                                                   |                    |          |                    |       |                                                                  |
|---------------------------------------------------------|--------------------|---------------------------------------------------|--------------------|----------|--------------------|-------|------------------------------------------------------------------|
| Param No.                                               | Sym.               | Characteristic                                    | Min.               | Typ†     | Max.               | Units | Conditions                                                       |
| <b>Data EEPROM Memory Specifications</b>                |                    |                                                   |                    |          |                    |       |                                                                  |
| MEM20                                                   | E <sub>D</sub>     | DataEE Byte Endurance                             | 100k               | —        | —                  | E/W   | -40°C ≤ T <sub>A</sub> ≤ +85°C                                   |
| MEM21                                                   | T <sub>D_RET</sub> | Characteristic Retention                          | —                  | 40       | —                  | Year  | Provided no other specifications are violated                    |
| MEM22                                                   | N <sub>D_REF</sub> | Total Erase/Write Cycles before Refresh           | 1M<br>500k         | 10M<br>— | —<br>—             | E/W   | -40°C ≤ T <sub>A</sub> ≤ +60°C<br>-40°C ≤ T <sub>A</sub> ≤ +85°C |
| MEM23                                                   | V <sub>D_RW</sub>  | V <sub>DD</sub> for Read or Erase/Write operation | V <sub>DDMIN</sub> | —        | V <sub>DDMAX</sub> | V     |                                                                  |
| MEM24                                                   | T <sub>D_BEW</sub> | Byte Erase and Write Cycle Time                   | —                  | 4.0      | 5.0                | ms    |                                                                  |
| <b>Program Flash Memory Specifications</b>              |                    |                                                   |                    |          |                    |       |                                                                  |
| MEM30                                                   | E <sub>P</sub>     | Memory Cell Endurance                             | 10k                | —        | —                  | E/W   | -40°C ≤ T <sub>A</sub> ≤ +85°C<br>(Note 1)                       |
| MEM32                                                   | T <sub>P_RET</sub> | Characteristic Retention                          | —                  | 40       | —                  | Year  | Provided no other specifications are violated                    |
| MEM33                                                   | V <sub>P_RD</sub>  | V <sub>DD</sub> for Read operation                | V <sub>DDMIN</sub> | —        | V <sub>DDMAX</sub> | V     |                                                                  |
| MEM34                                                   | V <sub>P_REW</sub> | V <sub>DD</sub> for Row Erase or Write operation  | V <sub>DDMIN</sub> | —        | V <sub>DDMAX</sub> | V     |                                                                  |
| MEM35                                                   | T <sub>P_REW</sub> | Self-Timed Row Erase or Self-Timed Write          | —                  | 2.0      | 2.5                | ms    |                                                                  |

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** Memory Cell Endurance for the Program memory is defined as: One Row Erase operation and one Self-Timed Write.

**TABLE 45-6: THERMAL CHARACTERISTICS**

**Standard Operating Conditions** (unless otherwise stated)

| Param No. | Sym.             | Characteristic                         | Typ. | Units | Conditions                                                                       |
|-----------|------------------|----------------------------------------|------|-------|----------------------------------------------------------------------------------|
| TH01      | $\theta_{JA}$    | Thermal Resistance Junction to Ambient | 60   | °C/W  | 28-pin SPDIP package                                                             |
|           |                  |                                        | 80   | °C/W  | 28-pin SOIC package                                                              |
|           |                  |                                        | 90   | °C/W  | 28-pin SSOP package                                                              |
|           |                  |                                        | 27.5 | °C/W  | 28-pin UQFN 4x4 mm package                                                       |
|           |                  |                                        | 27.5 | °C/W  | 28-pin QFN 6x6mm package                                                         |
| TH02      | $\theta_{JC}$    | Thermal Resistance Junction to Case    | 31.4 | °C/W  | 28-pin SPDIP package                                                             |
|           |                  |                                        | 24   | °C/W  | 28-pin SOIC package                                                              |
|           |                  |                                        | 24   | °C/W  | 28-pin SSOP package                                                              |
|           |                  |                                        | 24   | °C/W  | 28-pin UQFN 4x4mm package                                                        |
|           |                  |                                        | 24   | °C/W  | 28-pin QFN 6x6mm package                                                         |
| TH03      | TJMAX            | Maximum Junction Temperature           | 150  | °C    |                                                                                  |
| TH04      | PD               | Power Dissipation                      | —    | W     | $PD = P_{INTERNAL} + P_{I/O}^{(3)}$                                              |
| TH05      | PINTERNAL        | Internal Power Dissipation             | —    | W     | $P_{INTERNAL} = I_{DD} \times V_{DD}^{(1)}$                                      |
| TH06      | P <sub>I/O</sub> | I/O Power Dissipation                  | —    | W     | $P_{I/O} = \sum (I_{OL} \times V_{OL}) + \sum (I_{OH} \times (V_{DD} - V_{OH}))$ |
| TH07      | PDER             | Derated Power                          | —    | W     | $P_{DER} = P_{D_{MAX}} (T_J - T_A) / \theta_{JA}^{(2)}$                          |

**Note 1:** I<sub>DD</sub> is current to run the chip alone without driving any load on the output pins.

**2:** T<sub>A</sub> = Ambient Temperature, T<sub>J</sub> = Junction Temperature

**3:** See absolute maximum ratings for total power dissipation.

## 45.4 AC Characteristics

**FIGURE 45-4: LOAD CONDITIONS**

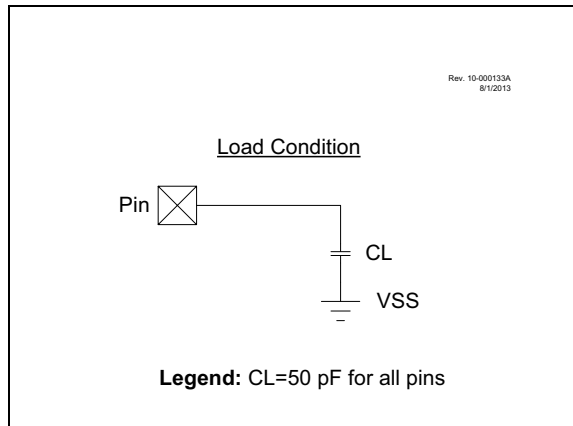


**FIGURE 45-5: CLOCK TIMING**

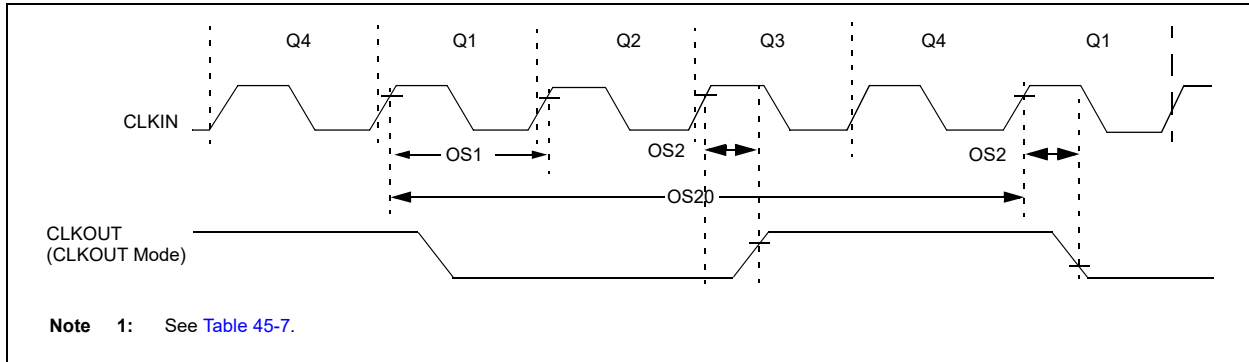


**TABLE 45-7: EXTERNAL CLOCK/OSCILLATOR TIMING REQUIREMENTS**

| Standard Operating Conditions (unless otherwise stated) |               |                        |      |        |      |       |                  |
|---------------------------------------------------------|---------------|------------------------|------|--------|------|-------|------------------|
| Param No.                                               | Sym.          | Characteristic         | Min. | Typ†   | Max. | Units | Conditions       |
| <b>ECL Oscillator</b>                                   |               |                        |      |        |      |       |                  |
| OS1                                                     | $F_{ECL}$     | Clock Frequency        | —    | —      | 500  | kHz   |                  |
| OS2                                                     | $T_{ECL\_DC}$ | Clock Duty Cycle       | 40   | —      | 60   | %     |                  |
| <b>ECM Oscillator</b>                                   |               |                        |      |        |      |       |                  |
| OS3                                                     | $F_{ECM}$     | Clock Frequency        | —    | —      | 4    | MHz   |                  |
| OS4                                                     | $T_{ECM\_DC}$ | Clock Duty Cycle       | 40   | —      | 60   | %     |                  |
| <b>ECH Oscillator</b>                                   |               |                        |      |        |      |       |                  |
| OS5                                                     | $F_{ECH}$     | Clock Frequency        | —    | —      | 64   | MHz   |                  |
| OS6                                                     | $T_{ECH\_DC}$ | Clock Duty Cycle       | 40   | —      | 60   | %     |                  |
| <b>LP Oscillator</b>                                    |               |                        |      |        |      |       |                  |
| OS7                                                     | $F_{LP}$      | Clock Frequency        | —    | —      | 100  | kHz   | Note 4           |
| <b>XT Oscillator</b>                                    |               |                        |      |        |      |       |                  |
| OS8                                                     | $F_{XT}$      | Clock Frequency        | —    | —      | 4    | MHz   | Note 4           |
| <b>HS Oscillator</b>                                    |               |                        |      |        |      |       |                  |
| OS9                                                     | $F_{HS}$      | Clock Frequency        | —    | —      | 20   | MHz   | Note 4           |
| <b>Secondary Oscillator</b>                             |               |                        |      |        |      |       |                  |
| OS10                                                    | $F_{SEC}$     | Clock Frequency        | 32.4 | 32.768 | 33.1 | kHz   |                  |
| <b>System Oscillator</b>                                |               |                        |      |        |      |       |                  |
| OS20                                                    | $F_{OSC}$     | System Clock Frequency | —    | —      | 64   | MHz   | (Note 2, Note 3) |

\* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

- Note 1:** Instruction cycle period ( $T_{CY}$ ) equals four times the input oscillator time base period. All specified values are based on characterization data for that particular oscillator type under standard operating conditions with the device executing code. Exceeding these specified limits may result in an unstable oscillator operation and/or higher than expected current consumption. All devices are tested to operate at "min" values with an external clock applied to OSC1 pin. When an external clock input is used, the "max" cycle time limit is "DC" (no clock) for all devices.
- 2:** The system clock frequency ( $F_{OSC}$ ) is selected by the "main clock switch controls" as described in [Section 10.0 "Power-Saving Operation Modes"](#).
- 3:** The system clock frequency ( $F_{OSC}$ ) must meet the voltage requirements defined in the [Section 45.2 "Standard Operating Conditions"](#).
- 4:** LP, XT and HS oscillator modes require an appropriate crystal or resonator to be connected to the device. For clocking the device with the external square wave, one of the EC mode selections must be used.

**TABLE 45-7: EXTERNAL CLOCK/OSCILLATOR TIMING REQUIREMENTS (CONTINUED)**

| Standard Operating Conditions (unless otherwise stated) |                 |                       |      |                     |      |       |            |
|---------------------------------------------------------|-----------------|-----------------------|------|---------------------|------|-------|------------|
| Param No.                                               | Sym.            | Characteristic        | Min. | Typ†                | Max. | Units | Conditions |
| OS21                                                    | F <sub>CY</sub> | Instruction Frequency | —    | F <sub>OSC</sub> /4 | —    | MHz   |            |
| OS22                                                    | T <sub>CY</sub> | Instruction Period    | 62.5 | 1/F <sub>CY</sub>   | —    | ns    |            |

\* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

- Note 1:** Instruction cycle period (T<sub>CY</sub>) equals four times the input oscillator time base period. All specified values are based on characterization data for that particular oscillator type under standard operating conditions with the device executing code. Exceeding these specified limits may result in an unstable oscillator operation and/or higher than expected current consumption. All devices are tested to operate at "min" values with an external clock applied to OSC1 pin. When an external clock input is used, the "max" cycle time limit is "DC" (no clock) for all devices.
- 2:** The system clock frequency (F<sub>OSC</sub>) is selected by the "main clock switch controls" as described in [Section 10.0 "Power-Saving Operation Modes"](#).
- 3:** The system clock frequency (F<sub>OSC</sub>) must meet the voltage requirements defined in the [Section 45.2 "Standard Operating Conditions"](#).
- 4:** LP, XT and HS oscillator modes require an appropriate crystal or resonator to be connected to the device. For clocking the device with the external square wave, one of the EC mode selections must be used.

**TABLE 45-8: INTERNAL OSCILLATOR PARAMETERS<sup>(1)</sup>**

| Standard Operating Conditions (unless otherwise stated) |          |                                           |              |                                |              |            |                          |
|---------------------------------------------------------|----------|-------------------------------------------|--------------|--------------------------------|--------------|------------|--------------------------|
| Param No.                                               | Sym.     | Characteristic                            | Min.         | Typ†                           | Max.         | Units      | Conditions               |
| OS50                                                    | FHFOSC   | Precision Calibrated HFINTOSC Frequency   | —            | 4<br>8<br>12<br>16<br>48<br>64 | —            | MHz        | (Note 2)                 |
| OS51                                                    | FHFOSCLP | Low-Power Optimized HFINTOSC Frequency    | 0.93<br>1.86 | 1<br>2                         | 1.07<br>2.14 | MHz<br>MHz |                          |
| OS53*                                                   | FLFOSC   | Internal LFINTOSC Frequency               | —            | 31                             | —            | kHz        |                          |
| OS54*                                                   | THFOSCST | HFINTOSC Wake-up from Sleep Start-up Time | —<br>—       | 11<br>50                       | 20<br>—      | μs<br>μs   | VREGPM = 0<br>VREGPM = 1 |
| OS56                                                    | TLFOSCST | LFINTOSC Wake-up from Sleep Start-up Time | —            | 0.2                            | —            | ms         |                          |

\* These parameters are characterized but not tested.

† Data in “Typ” column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** To ensure these oscillator frequency tolerances, VDD and VSS must be capacitively decoupled as close to the device as possible. 0.1 μF and 0.01 μF values in parallel are recommended.

**2:** See [Figure 45-6: Precision Calibrated HFINTOSC Frequency Accuracy Over Device VDD and Temperature](#).

**FIGURE 45-6: PRECISION CALIBRATED HFINTOSC FREQUENCY ACCURACY OVER DEVICE VDD AND TEMPERATURE**

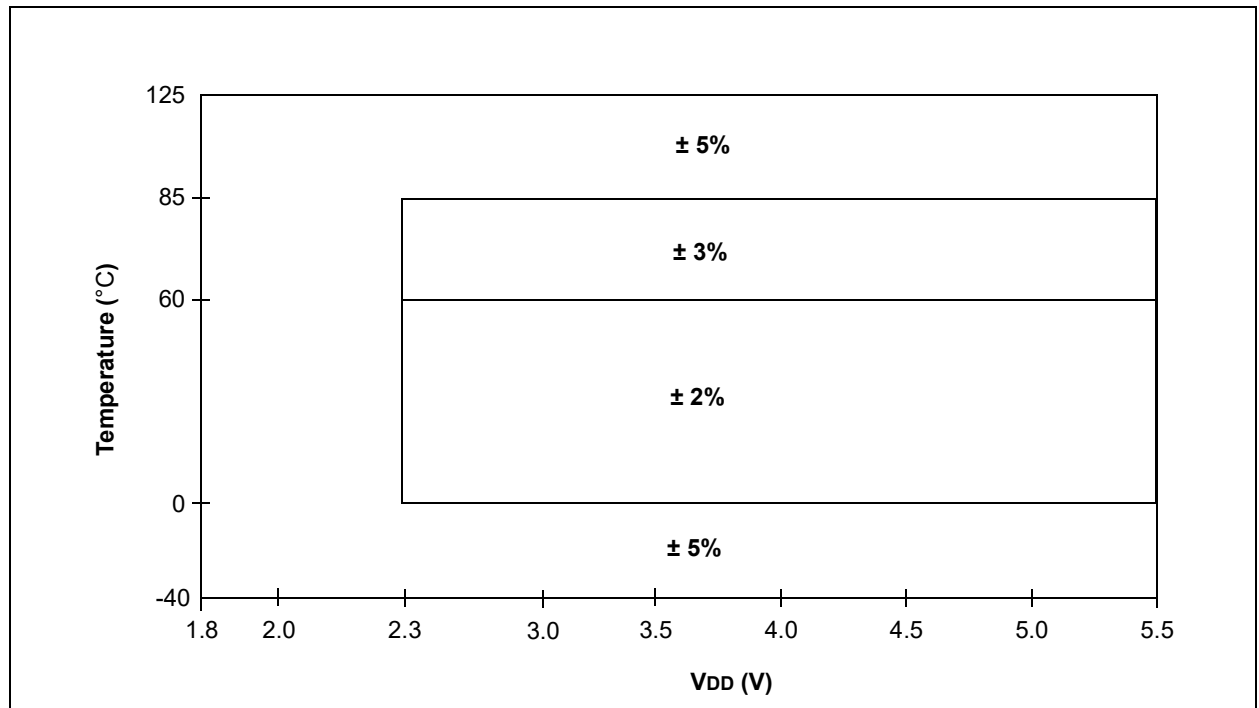


**TABLE 45-9: PLL SPECIFICATIONS**

| <b>Standard Operating Conditions</b> (unless otherwise stated) $V_{DD} \geq 2.5V$ |         |                                         |       |      |      |         |               |
|-----------------------------------------------------------------------------------|---------|-----------------------------------------|-------|------|------|---------|---------------|
| Param No.                                                                         | Sym.    | Characteristic                          | Min.  | Typ† | Max. | Units   | Conditions    |
| PLL01                                                                             | FPLLIN  | PLL Input Frequency Range               | 4     | —    | 16   | MHz     |               |
| PLL02                                                                             | FPLLOUT | PLL Output Frequency Range              | 16    | —    | 64   | MHz     | <b>Note 1</b> |
| PLL03                                                                             | TPLLST  | PLL Lock Time from Start-up             | —     | 200  | —    | $\mu s$ |               |
| PLL04                                                                             | FPLLJIT | PLL Output Frequency Stability (Jitter) | -0.25 | —    | 0.25 | %       |               |

\* These parameters are characterized but not tested.

† Data in “Typ” column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** The output frequency of the PLL must meet the FOSC requirements listed in Parameter D002.



**FIGURE 45-7: CLKOUT AND I/O TIMING**

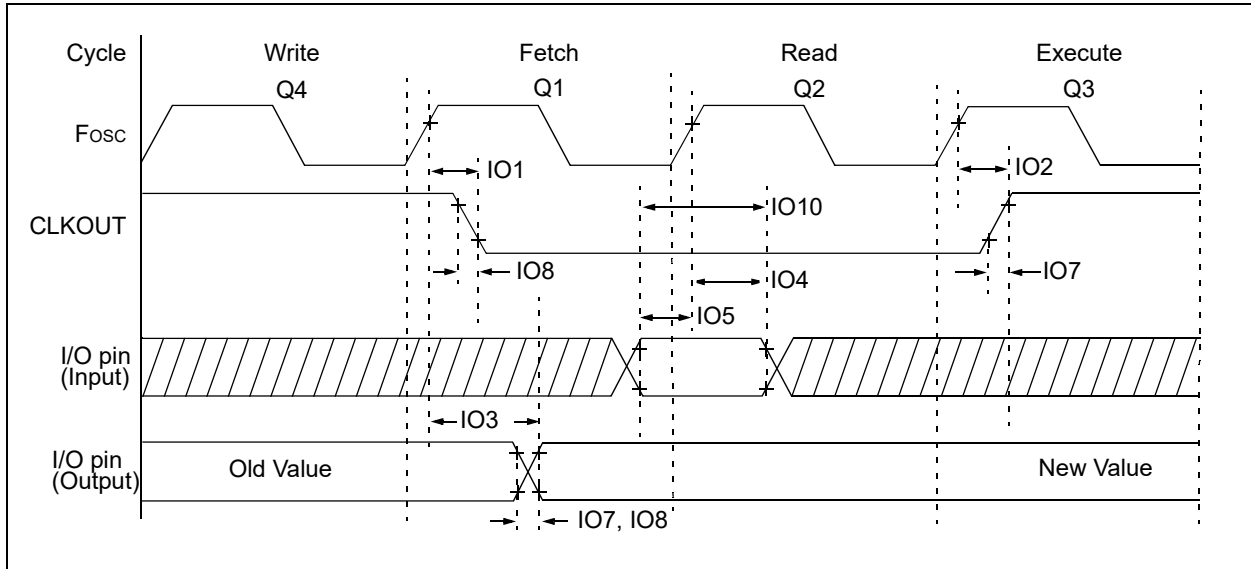


**TABLE 45-10: I/O AND CLKOUT TIMING SPECIFICATIONS**

| Standard Operating Conditions (unless otherwise stated) |                   |                                                                               |      |      |      |       |                 |
|---------------------------------------------------------|-------------------|-------------------------------------------------------------------------------|------|------|------|-------|-----------------|
| Param No.                                               | Sym.              | Characteristic                                                                | Min. | Typ† | Max. | Units | Conditions      |
| IO1*                                                    | $T_{CLKOUTH}$     | CLKOUT rising edge delay (rising edge Fosc (Q1 cycle) to falling edge CLKOUT) | —    | —    | 70   | ns    |                 |
| IO2*                                                    | $T_{CLKOUTL}$     | CLKOUT falling edge delay (rising edge Fosc (Q3 cycle) to rising edge CLKOUT) | —    | —    | 72   | ns    |                 |
| IO3*                                                    | $T_{IO\_VALID}$   | Port output valid time (rising edge Fosc (Q1 cycle) to port valid)            | —    | 50   | 70   | ns    |                 |
| IO4*                                                    | $T_{IO\_SETUP}$   | Port input setup time (Setup time before rising edge Fosc – Q2 cycle)         | 20   | —    | —    | ns    |                 |
| IO5*                                                    | $T_{IO\_HOLD}$    | Port input hold time (Hold time after rising edge Fosc – Q2 cycle)            | 50   | —    | —    | ns    |                 |
| IO6*                                                    | $T_{IOR\_SLREN}$  | Port I/O rise time, slew rate enabled                                         | —    | 25   | —    | ns    | $V_{DD} = 3.0V$ |
| IO7*                                                    | $T_{IOR\_SLRDIS}$ | Port I/O rise time, slew rate disabled                                        | —    | 5    | —    | ns    | $V_{DD} = 3.0V$ |
| IO8*                                                    | $T_{IOF\_SLREN}$  | Port I/O fall time, slew rate enabled                                         | —    | 25   | —    | ns    | $V_{DD} = 3.0V$ |
| IO9*                                                    | $T_{IOF\_SLRDIS}$ | Port I/O fall time, slew rate disabled                                        | —    | 5    | —    | ns    | $V_{DD} = 3.0V$ |
| IO10*                                                   | $T_{INT}$         | INT pin high or low time to trigger an interrupt                              | 25   | —    | —    | ns    |                 |
| IO11*                                                   | $T_{IOC}$         | Interrupt-on-Change minimum high or low time to trigger interrupt             | 25   | —    | —    | ns    |                 |

\*These parameters are characterized but not tested.

**FIGURE 45-8: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER TIMING**

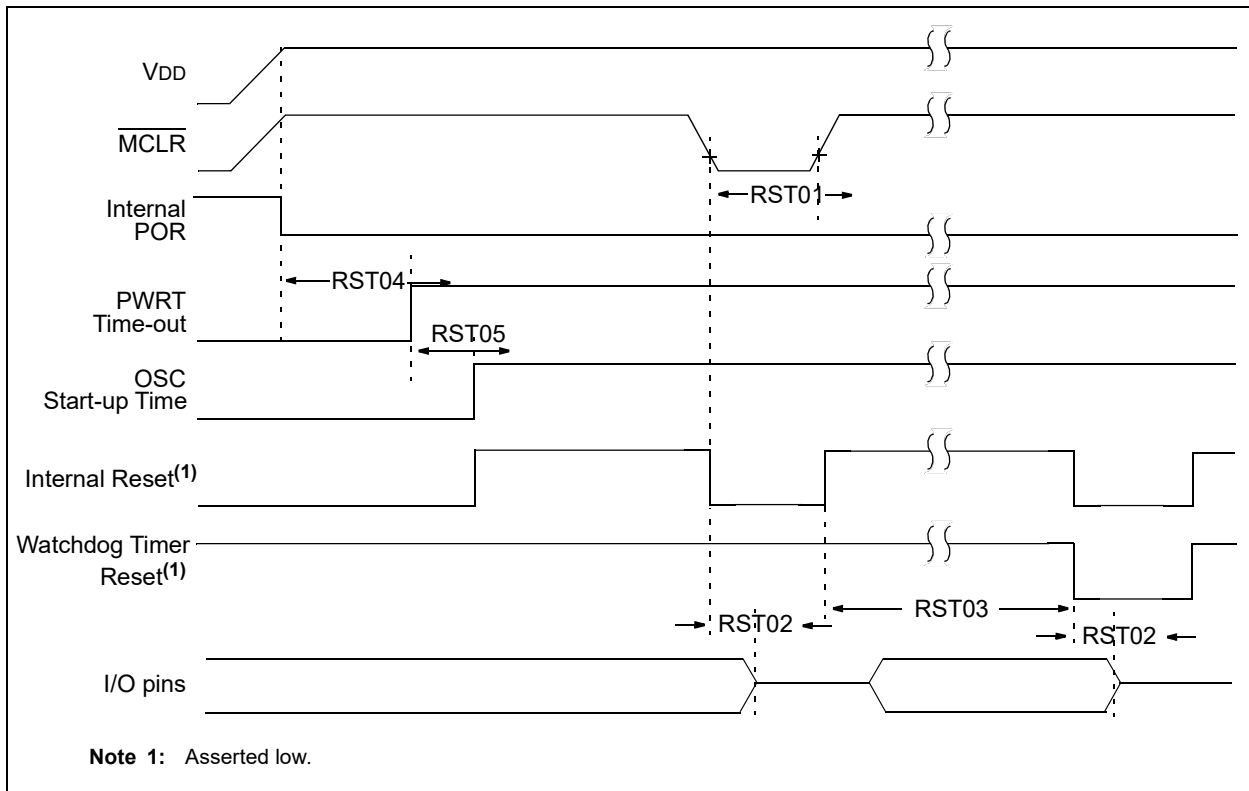


**FIGURE 45-9: BROWN-OUT RESET TIMING AND CHARACTERISTICS**

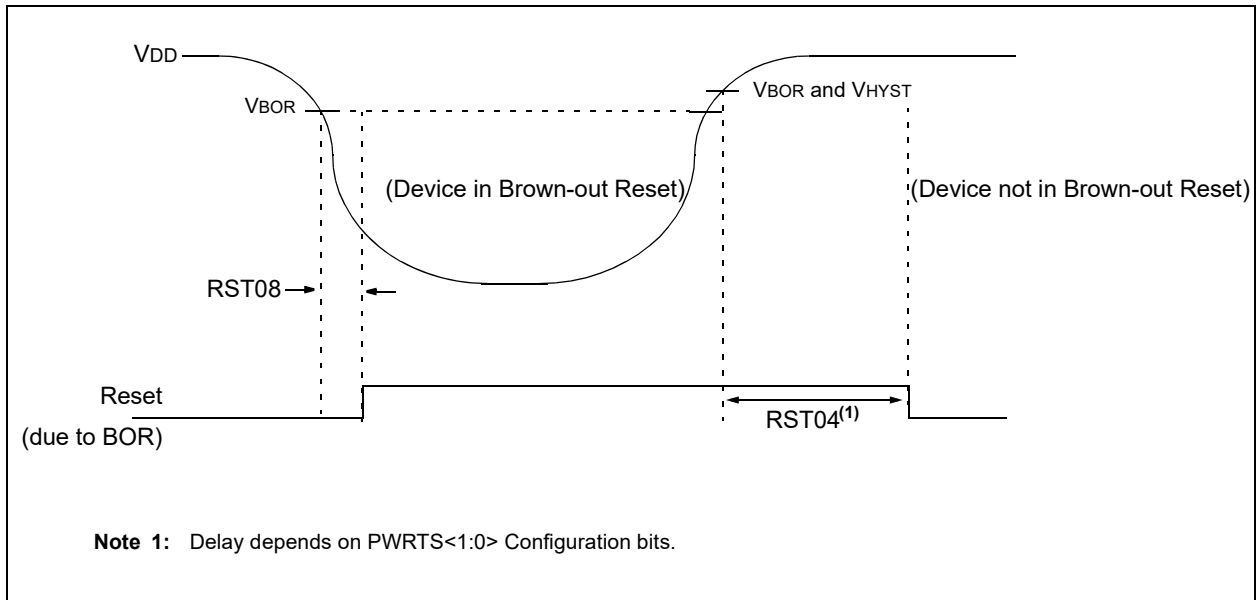


**TABLE 45-11: RESET, WDT, OSCILLATOR START-UP TIMER, POWER-UP TIMER, BROWN-OUT RESET AND LOW-POWER BROWN-OUT RESET SPECIFICATIONS**

| Standard Operating Conditions (unless otherwise stated) |         |                                                   |                                  |                                    |                                   |                       |                                                                                        |
|---------------------------------------------------------|---------|---------------------------------------------------|----------------------------------|------------------------------------|-----------------------------------|-----------------------|----------------------------------------------------------------------------------------|
| Param No.                                               | Sym.    | Characteristic                                    | Min.                             | Typ†                               | Max.                              | Units                 | Conditions                                                                             |
| RST01*                                                  | TMCLR   | MCLR Pulse Width Low to ensure Reset              | 2                                | —                                  | —                                 | μs                    |                                                                                        |
| RST02*                                                  | TIOZ    | I/O high-impedance from Reset detection           | —                                | —                                  | 2                                 | μs                    |                                                                                        |
| RST03                                                   | TWDT    | Watchdog Timer Time-out Period                    | —                                | 16                                 | —                                 | ms                    | 1:512 Prescaler                                                                        |
| RST04*                                                  | TPWRT   | Power-up Timer Period                             | —                                | 1<br>16<br>64                      | —                                 | ms<br>ms<br>ms        | PWRTS = 00<br>PWRTS = 01<br>PWRTS = 10                                                 |
| RST05                                                   | TOST    | Oscillator Start-up Timer Period <sup>(1,2)</sup> | —                                | 1024                               | —                                 | Tosc                  |                                                                                        |
| RST06                                                   | VBOR    | Brown-out Reset Voltage <sup>(4)</sup>            | 2.7<br>2.55<br>2.3<br>2.3<br>1.8 | 2.85<br>2.7<br>2.45<br>2.45<br>1.9 | 3.0<br>2.85<br>2.6<br>2.6<br>2.05 | V<br>V<br>V<br>V<br>V | BORV = 00<br>BORV = 01<br>BORV = 10<br>BORV = 11 (PIC18Fxxx)<br>BORV = 11 (PIC18LFxxx) |
| RST07                                                   | VBORHYS | Brown-out Reset Hysteresis                        | —                                | 40                                 | —                                 | mV                    |                                                                                        |
| RST08                                                   | TBORDC  | Brown-out Reset Response Time                     | —                                | 3                                  | —                                 | μs                    |                                                                                        |
| RST09                                                   | VLPBOR  | Low-Power Brown-out Reset Voltage                 | 1.8                              | 1.9                                | 2.2                               | V                     |                                                                                        |

\* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

- Note 1:** By design, the Oscillator Start-up Timer (OST) counts the first 1024 cycles, independent of frequency.
- Note 2:** To ensure these voltage tolerances, VDD and VSS must be capacitively decoupled as close to the device as possible. 0.1 μF and 0.01 μF values in parallel are recommended.

**TABLE 45-12: HIGH/LOW-VOLTAGE DETECT CHARACTERISTICS**

| Standard Operating Conditions (unless otherwise stated) |                  |                   |      |      |      |       |               |
|---------------------------------------------------------|------------------|-------------------|------|------|------|-------|---------------|
| Param. No.                                              | Symbol           | Characteristic    | Min. | Typ† | Max. | Units | Conditions    |
| HLVD01                                                  | V <sub>DET</sub> | Voltage Detection | —    | 1.90 | —    | V     | SEL<3:0>=0000 |
|                                                         |                  |                   | —    | 2.10 | —    | V     | SEL<3:0>=0001 |
|                                                         |                  |                   | —    | 2.25 | —    | V     | SEL<3:0>=0010 |
|                                                         |                  |                   | —    | 2.50 | —    | V     | SEL<3:0>=0011 |
|                                                         |                  |                   | —    | 2.60 | —    | V     | SEL<3:0>=0100 |
|                                                         |                  |                   | —    | 2.75 | —    | V     | SEL<3:0>=0101 |
|                                                         |                  |                   | —    | 2.90 | —    | V     | SEL<3:0>=0110 |
|                                                         |                  |                   | —    | 3.15 | —    | V     | SEL<3:0>=0111 |
|                                                         |                  |                   | —    | 3.35 | —    | V     | SEL<3:0>=1000 |
|                                                         |                  |                   | —    | 3.60 | —    | V     | SEL<3:0>=1001 |
|                                                         |                  |                   | —    | 3.75 | —    | V     | SEL<3:0>=1010 |
|                                                         |                  |                   | —    | 4.00 | —    | V     | SEL<3:0>=1011 |
|                                                         |                  |                   | —    | 4.20 | —    | V     | SEL<3:0>=1100 |
|                                                         |                  |                   | —    | 4.35 | —    | V     | SEL<3:0>=1101 |
|                                                         |                  |                   | —    | 4.65 | —    | V     | SEL<3:0>=1110 |

**TABLE 45-13: ANALOG-TO-DIGITAL CONVERTER (ADC) ACCURACY SPECIFICATIONS<sup>(1,2)</sup>:**

| <b>Operating Conditions</b> (unless otherwise stated)<br>V <sub>DD</sub> = 3.0V, T <sub>A</sub> = 25°C, T <sub>AD</sub> = 1μs |        |                                                |        |      |                 |       |                                                        |
|-------------------------------------------------------------------------------------------------------------------------------|--------|------------------------------------------------|--------|------|-----------------|-------|--------------------------------------------------------|
| Param No.                                                                                                                     | Sym.   | Characteristic                                 | Min.   | Typ† | Max.            | Units | Conditions                                             |
| AD01                                                                                                                          | NR     | Resolution                                     | —      | —    | 12              | bit   |                                                        |
| AD02                                                                                                                          | EIL    | Integral Error                                 | —      | ±0.1 | ±2.0            | LSb   | ADCRE <sub>F</sub> + = 3.0V, ADCRE <sub>F</sub> - = 0V |
| AD03                                                                                                                          | EDL    | Differential Error                             | —      | ±0.1 | ±1.0            | LSb   | ADCRE <sub>F</sub> + = 3.0V, ADCRE <sub>F</sub> - = 0V |
| AD04                                                                                                                          | EOFF   | Offset Error                                   | —      | 0.5  | 6.0             | LSb   | ADCRE <sub>F</sub> + = 3.0V, ADCRE <sub>F</sub> - = 0V |
| AD05                                                                                                                          | EGN    | Gain Error                                     | —      | ±0.2 | ±6.0            | LSb   | ADCRE <sub>F</sub> + = 3.0V, ADCRE <sub>F</sub> - = 0V |
| AD06                                                                                                                          | VADREF | ADC Reference Voltage (ADREF+ - ADREF-)        | 1.8    | —    | V <sub>DD</sub> | V     |                                                        |
| AD07                                                                                                                          | VAIN   | Full-Scale Range                               | ADREF- | —    | ADREF+          | V     |                                                        |
| AD08                                                                                                                          | ZAIN   | Recommended Impedance of Analog Voltage Source | —      | 10   | —               | kΩ    |                                                        |
| AD09                                                                                                                          | RVREF  | ADC Voltage Reference Ladder Impedance         | —      | 50   | —               | kΩ    | <b>Note 3</b>                                          |

\* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** Total Absolute Error is the sum of the offset, gain and integral non-linearity (INL) errors.

**2:** The ADC conversion result never decreases with an increase in the input and has no missing codes.

**3:** This is the impedance seen by the VREF pads when the external reference pads are selected.

**TABLE 45-14: ANALOG-TO-DIGITAL CONVERTER (ADC) CONVERSION TIMING SPECIFICATIONS**

| Standard Operating Conditions (unless otherwise stated) |      |                                           |      |              |      |       |                                              |
|---------------------------------------------------------|------|-------------------------------------------|------|--------------|------|-------|----------------------------------------------|
| Param No.                                               | Sym. | Characteristic                            | Min. | Typ†         | Max. | Units | Conditions                                   |
| AD20                                                    | TAD  | ADC Clock Period                          | 0.5  | —            | 9    | μs    | Using Fosc as the ADC clock source ADCS = 0  |
| AD21                                                    |      |                                           | —    | 2            | —    | μs    | Using ADCRC as the ADC clock source ADCS = 1 |
| AD22                                                    | TCNV | Conversion Time <sup>(1)</sup>            | —    | 14 TAD+2 TCY | —    | —     | Using FOSC as the ADC clock source ADCS = 1  |
|                                                         |      |                                           | —    | 16 TAD+2 TCY | —    | —     | Using ADCRC as the ADC clock source ADCS = 0 |
| AD24                                                    | THCD | Sample and Hold Capacitor Disconnect Time | —    | 2 TAD+1 TCY  | —    | —     | Using FOSC as the ADC clock source ADCS = 1  |
|                                                         |      |                                           | —    | 3 TAD+2 TCY  | —    | —     | Using ADCRC as the ADC clock source ADCS = 0 |

\* These parameters are characterized but not tested.

† Data in “Typ” column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** Does not apply for the ADCRC oscillator.

**FIGURE 45-10: ADC CONVERSION TIMING (ADC CLOCK Fosc-BASED)**

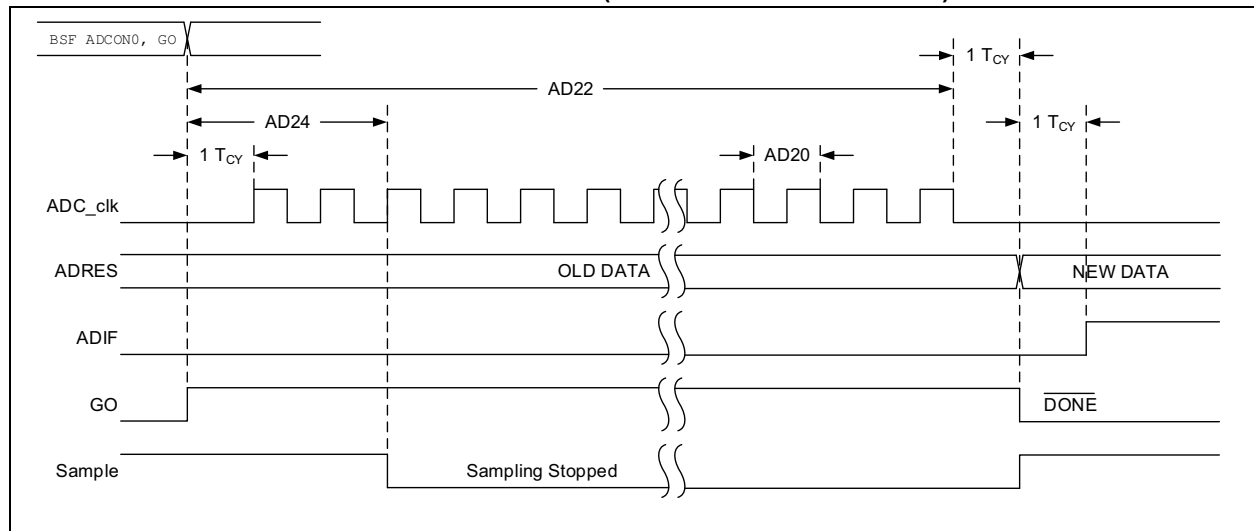


**FIGURE 45-11: ADC CONVERSION TIMING (ADC CLOCK FROM ADCRC)**

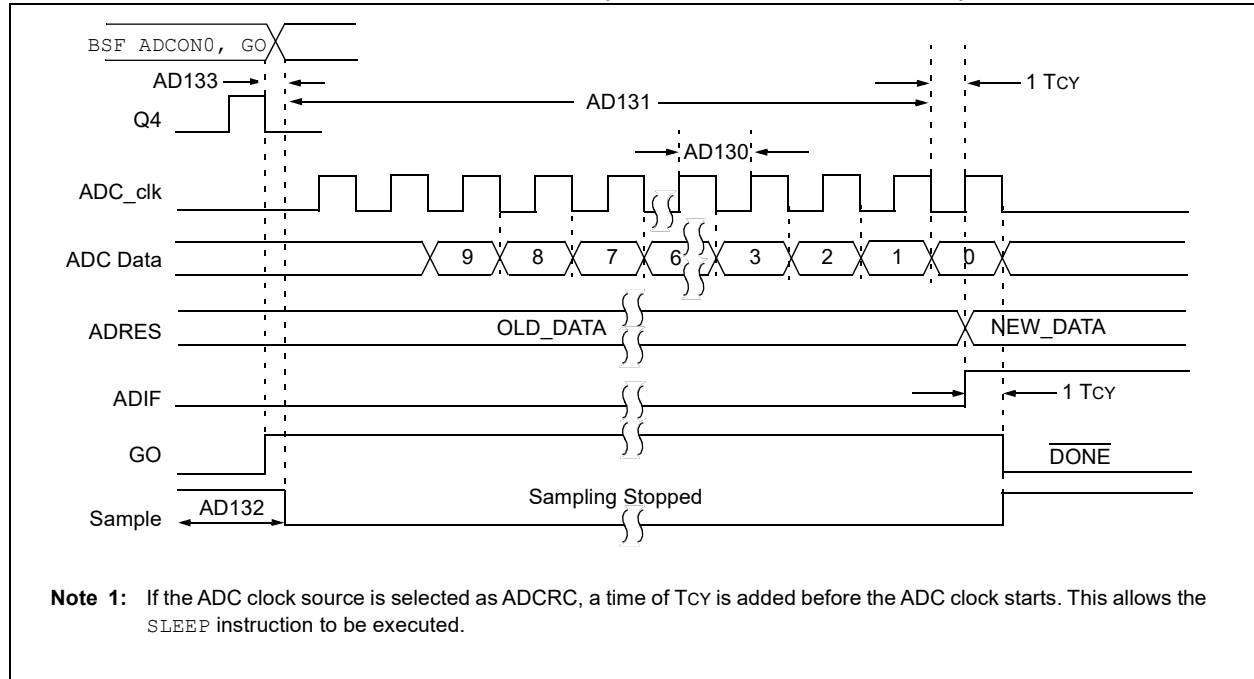


**TABLE 45-15: COMPARATOR SPECIFICATIONS**

| <b>Operating Conditions</b> (unless otherwise stated)<br>V <sub>DD</sub> = 3.0V, T <sub>A</sub> = 25°C |                                  |                                   |      |      |                 |       |                                       |
|--------------------------------------------------------------------------------------------------------|----------------------------------|-----------------------------------|------|------|-----------------|-------|---------------------------------------|
| Param No.                                                                                              | Sym.                             | Characteristics                   | Min. | Typ. | Max.            | Units | Comments                              |
| CM01                                                                                                   | V <sub>IOFF</sub>                | Input Offset Voltage              | —    | —    | ±40             | mV    | V <sub>ICM</sub> = V <sub>DD</sub> /2 |
| CM02                                                                                                   | V <sub>ICM</sub>                 | Input Common Mode Range           | GND  | —    | V <sub>DD</sub> | V     |                                       |
| CM03                                                                                                   | CMRR                             | Common Mode Input Rejection Ratio | —    | 50   | —               | dB    |                                       |
| CM04                                                                                                   | V <sub>HYST</sub>                | Comparator Hysteresis             | 10   | 25   | 40              | mV    |                                       |
| CM05                                                                                                   | T <sub>RESP</sub> <sup>(1)</sup> | Response Time, Rising Edge        | —    | 300  | 600             | ns    |                                       |
|                                                                                                        |                                  | Response Time, Falling Edge       | —    | 220  | 500             | ns    |                                       |

\* These parameters are characterized but not tested.

**Note 1:** Response time measured with one comparator input at V<sub>DD</sub>/2, while the other input transitions from V<sub>SS</sub> to V<sub>DD</sub>.

**2:** A mode change includes changing any of the control register values, including module enable.

**TABLE 45-16: 5-BIT DAC SPECIFICATIONS**

| <b>Standard Operating Conditions</b> (unless otherwise stated)<br>V <sub>DD</sub> = 3.0V, T <sub>A</sub> = 25°C |                   |                              |      |                                                     |       |       |          |
|-----------------------------------------------------------------------------------------------------------------|-------------------|------------------------------|------|-----------------------------------------------------|-------|-------|----------|
| Param No.                                                                                                       | Sym.              | Characteristics              | Min. | Typ.                                                | Max.  | Units | Comments |
| DSB01                                                                                                           | V <sub>LSB</sub>  | Step Size                    | —    | (V <sub>DACREF+</sub> - V <sub>DACREF-</sub> ) / 32 | —     | V     |          |
| DSB01                                                                                                           | V <sub>ACC</sub>  | Absolute Accuracy            | —    | —                                                   | ± 0.5 | LSb   |          |
| DSB03*                                                                                                          | R <sub>UNIT</sub> | Unit Resistor Value          | —    | 5000                                                | —     | Ω     |          |
| DSB04*                                                                                                          | T <sub>ST</sub>   | Settling Time <sup>(1)</sup> | —    | —                                                   | 10    | μs    |          |

\* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** Settling time measured while DACR<4:0> transitions from '00000' to '01111'.

**TABLE 45-17: FIXED VOLTAGE REFERENCE (FVR) SPECIFICATIONS**

| <b>Standard Operating Conditions</b> (unless otherwise stated) |                    |                   |      |      |      |       |                                        |
|----------------------------------------------------------------|--------------------|-------------------|------|------|------|-------|----------------------------------------|
| Param. No.                                                     | Symbol             | Characteristic    | Min. | Typ. | Max. | Units | Conditions                             |
| FVR01                                                          | V <sub>FVR1</sub>  | 1x Gain (1.024V)  | -4   | —    | +4   | %     | V <sub>DD</sub> ≥ 2.5V, -40°C to 85°C  |
| FVR02                                                          | V <sub>FVR2</sub>  | 2x Gain (2.048V)  | -4   | —    | +4   | %     | V <sub>DD</sub> ≥ 2.5V, -40°C to 85°C  |
| FVR03                                                          | V <sub>FVR4</sub>  | 4x Gain (4.096V)  | -5   | —    | +5   | %     | V <sub>DD</sub> ≥ 4.75V, -40°C to 85°C |
| FVR04                                                          | T <sub>FVRST</sub> | FVR Start-up Time | —    | 25   | —    | us    |                                        |

**TABLE 45-18: ZERO CROSS DETECT (ZCD) SPECIFICATIONS**

| <b>Standard Operating Conditions</b> (unless otherwise stated)<br>V <sub>DD</sub> = 3.0V, T <sub>A</sub> = 25°C |                      |                                |     |      |     |       |          |
|-----------------------------------------------------------------------------------------------------------------|----------------------|--------------------------------|-----|------|-----|-------|----------|
| Param. No.                                                                                                      | Sym.                 | Characteristics                | Min | Typ† | Max | Units | Comments |
| ZC01                                                                                                            | V <sub>PINZC</sub>   | Voltage on Zero Cross Pin      | —   | 0.75 | —   | V     |          |
| ZC02                                                                                                            | I <sub>ZCD_MAX</sub> | Maximum source or sink current | —   | —    | 600 | μA    |          |
| ZC03                                                                                                            | T <sub>RESPH</sub>   | Response Time, Rising Edge     | —   | 1    | —   | μs    |          |
|                                                                                                                 | T <sub>RESPL</sub>   | Response Time, Falling Edge    | —   | 1    | —   | μs    |          |

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**FIGURE 45-12: TIMER0 AND TIMER1 EXTERNAL CLOCK TIMINGS**

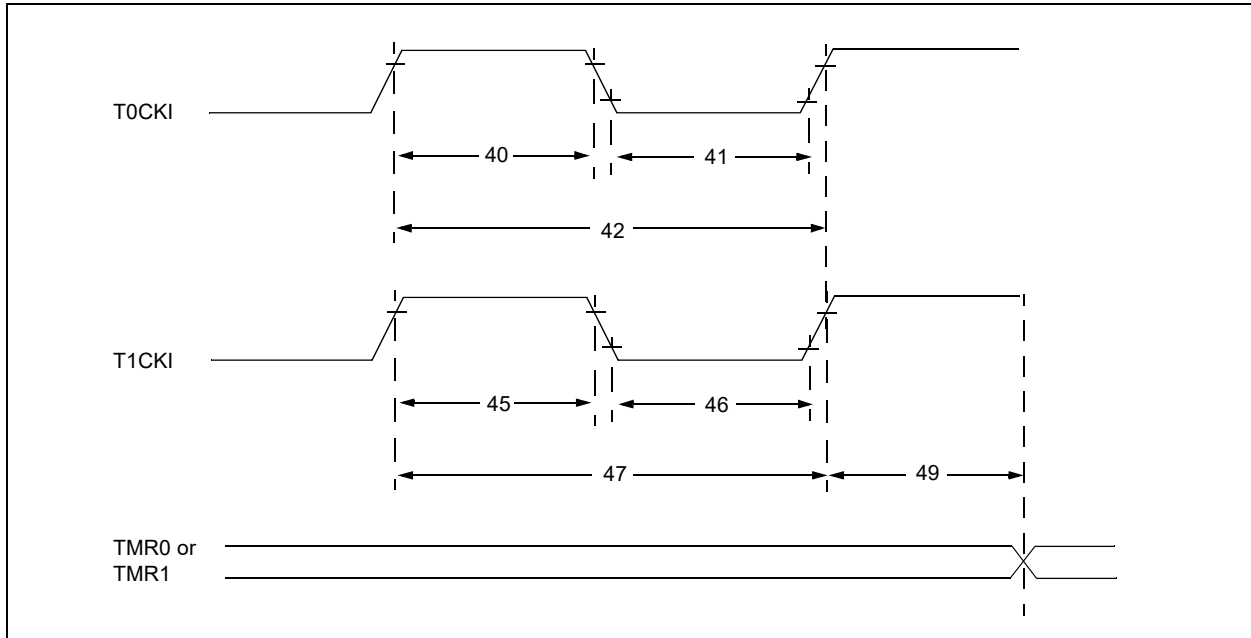


**TABLE 45-19: TIMER0 AND TIMER1 EXTERNAL CLOCK REQUIREMENTS**

| Standard Operating Conditions (unless otherwise stated)                        |           |                                                   |                             |                                                |      |             |       |                     |
|--------------------------------------------------------------------------------|-----------|---------------------------------------------------|-----------------------------|------------------------------------------------|------|-------------|-------|---------------------|
| Operating Temperature $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ |           |                                                   |                             |                                                |      |             |       |                     |
| Param No.                                                                      | Sym.      | Characteristic                                    |                             | Min.                                           | Typ† | Max.        | Units | Conditions          |
| 40*                                                                            | Tt0H      | T0CKI High Pulse Width                            | No Prescaler                | $0.5 T_{CY} + 20$                              | —    | —           | ns    |                     |
|                                                                                |           |                                                   | With Prescaler              | 10                                             | —    | —           | ns    |                     |
| 41*                                                                            | Tt0L      | T0CKI Low Pulse Width                             | No Prescaler                | $0.5 T_{CY} + 20$                              | —    | —           | ns    |                     |
|                                                                                |           |                                                   | With Prescaler              | 10                                             | —    | —           | ns    |                     |
| 42*                                                                            | Tt0P      | T0CKI Period                                      |                             | Greater of:<br>$20$ or $\frac{T_{CY} + 40}{N}$ | —    | —           | ns    | N = prescale value  |
| 45*                                                                            | Tt1H      | T1CKI High Time                                   | Synchronous, No Prescaler   | $0.5 T_{CY} + 20$                              | —    | —           | ns    |                     |
|                                                                                |           |                                                   | Synchronous, with Prescaler | 15                                             | —    | —           | ns    |                     |
|                                                                                |           |                                                   | Asynchronous                | 30                                             | —    | —           | ns    |                     |
| 46*                                                                            | Tt1L      | T1CKI Low Time                                    | Synchronous, No Prescaler   | $0.5 T_{CY} + 20$                              | —    | —           | ns    |                     |
|                                                                                |           |                                                   | Synchronous, with Prescaler | 15                                             | —    | —           | ns    |                     |
|                                                                                |           |                                                   | Asynchronous                | 30                                             | —    | —           | ns    |                     |
| 47*                                                                            | Tt1P      | T1CKI Input Period                                | Synchronous                 | Greater of:<br>$30$ or $\frac{T_{CY} + 40}{N}$ | —    | —           | ns    | N = prescale value  |
|                                                                                |           |                                                   | Asynchronous                | 60                                             | —    | —           | ns    |                     |
| 49*                                                                            | TCKEZTMR1 | Delay from External Clock Edge to Timer Increment |                             | $2 T_{OSC}$                                    | —    | $7 T_{OSC}$ | —     | Timers in Sync mode |

\* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.



**FIGURE 45-13: CAPTURE/COMPARE/PWM TIMINGS (CCP)**

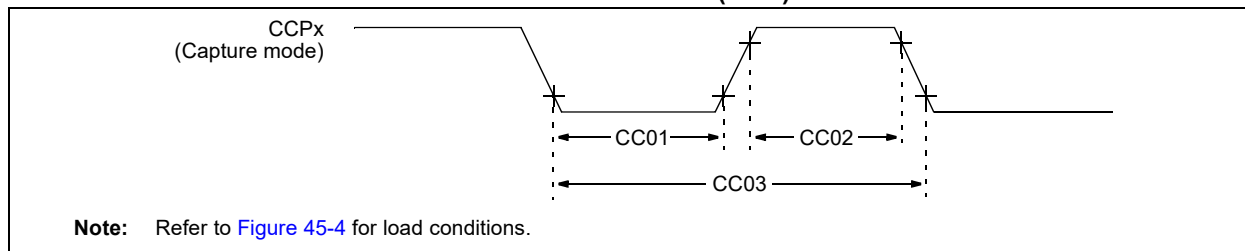


**TABLE 45-20: CAPTURE/COMPARE/PWM REQUIREMENTS (CCP)**

| Standard Operating Conditions (unless otherwise stated)                        |      |                      |                |                          |      |      |       |                    |
|--------------------------------------------------------------------------------|------|----------------------|----------------|--------------------------|------|------|-------|--------------------|
| Operating Temperature $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ |      |                      |                |                          |      |      |       |                    |
| Param No.                                                                      | Sym. | Characteristic       |                | Min.                     | Typ† | Max. | Units | Conditions         |
| CC01*                                                                          | TccL | CCPx Input Low Time  | No Prescaler   | $0.5T_{CY} + 20$         | —    | —    | ns    |                    |
|                                                                                |      |                      | With Prescaler | 20                       | —    | —    | ns    |                    |
| CC02*                                                                          | TccH | CCPx Input High Time | No Prescaler   | $0.5T_{CY} + 20$         | —    | —    | ns    |                    |
|                                                                                |      |                      | With Prescaler | 20                       | —    | —    | ns    |                    |
| CC03*                                                                          | TccP | CCPx Input Period    |                | $\frac{3T_{CY} + 40}{N}$ | —    | —    | ns    | N = prescale value |

\* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**FIGURE 45-14: SPI MASTER MODE TIMING (CKE = 0, SMP = 0)**

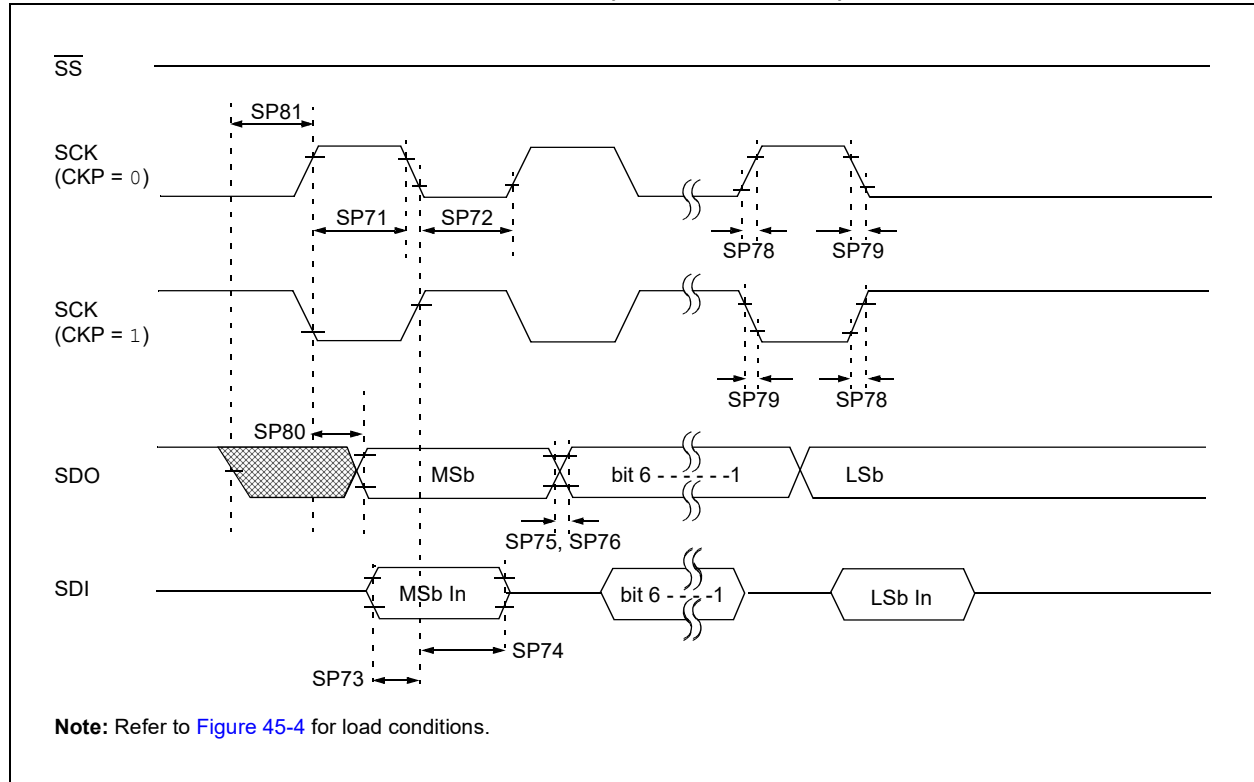


**FIGURE 45-15: SPI MASTER MODE TIMING (CKE = 1, SMP = 1)**

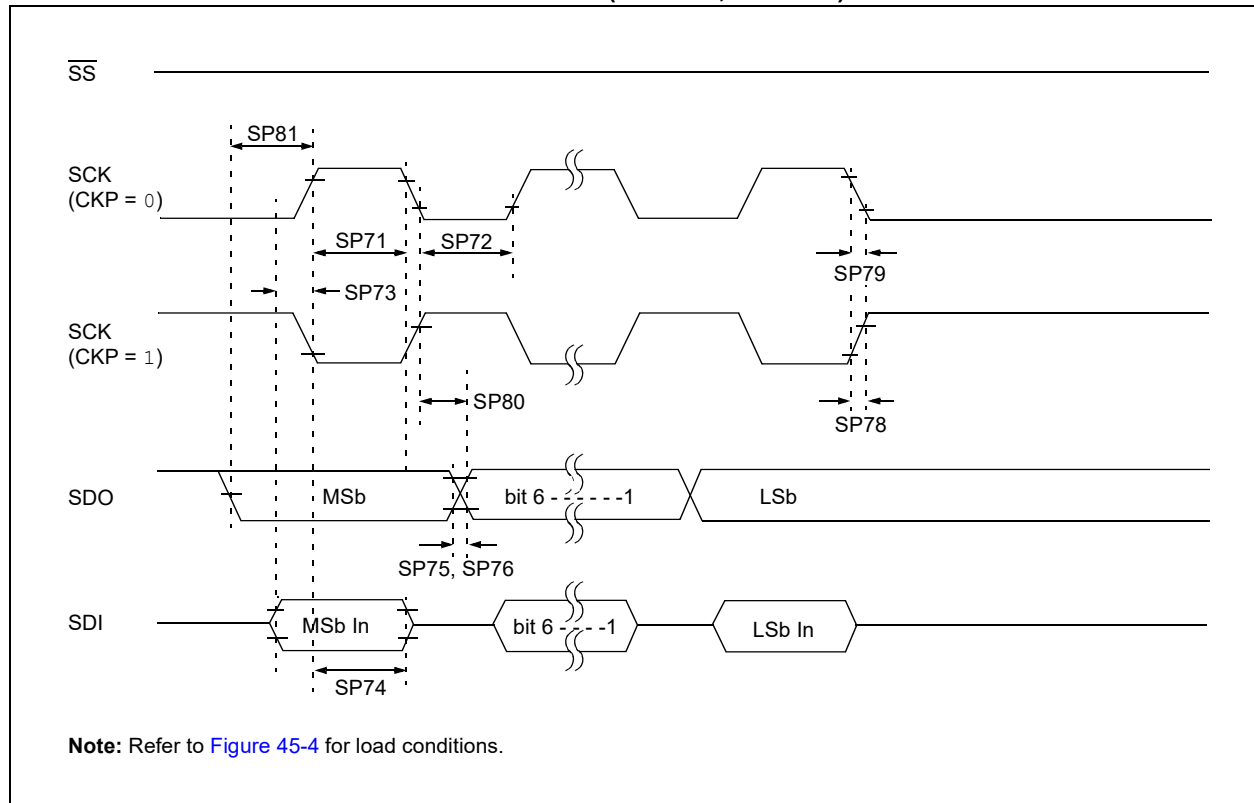


**FIGURE 45-16: SPI SLAVE MODE TIMING (CKE = 0)**

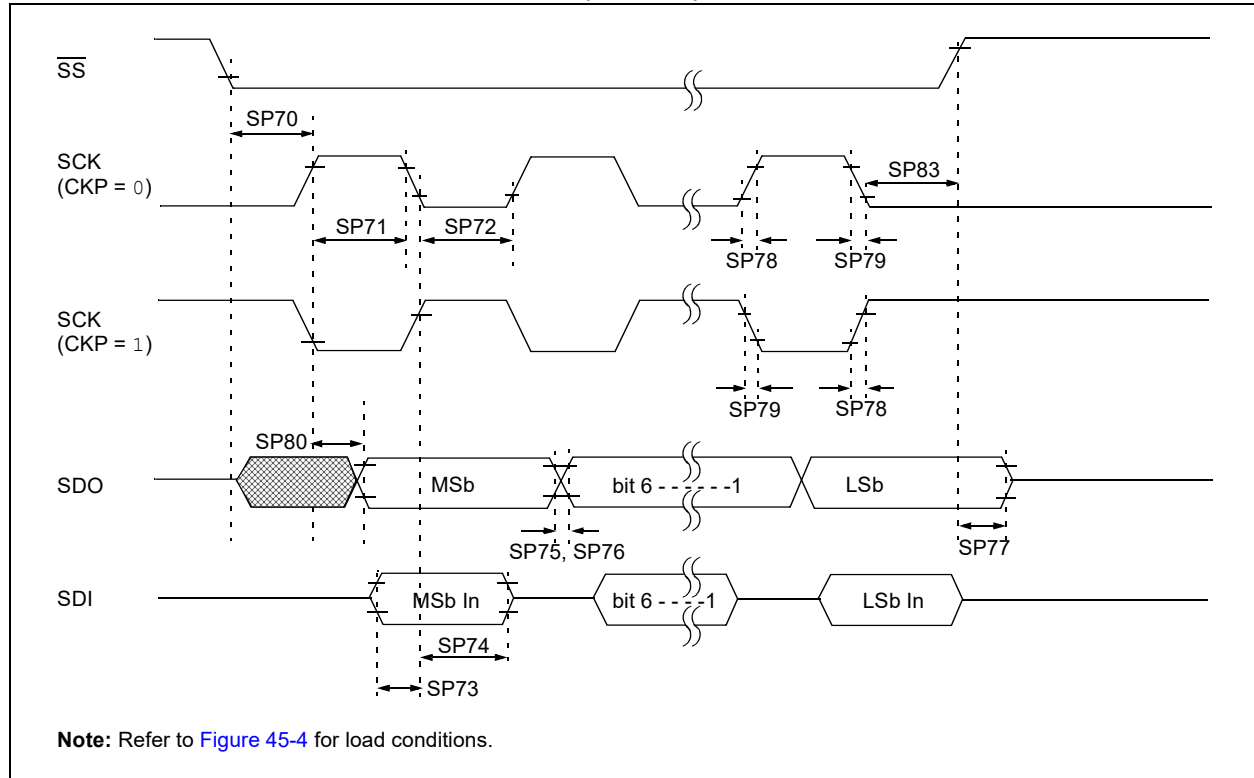


**FIGURE 45-17: SPI SLAVE MODE TIMING (CKE = 1)**

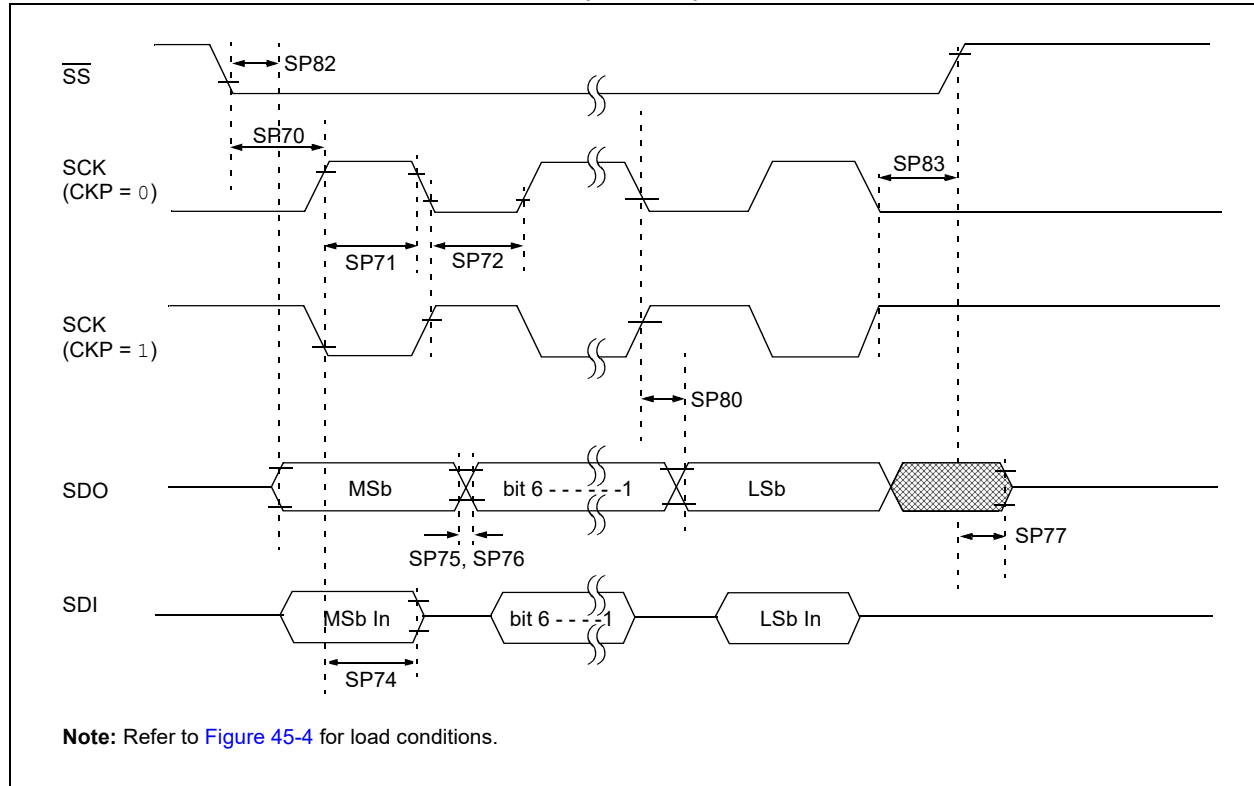


**TABLE 45-21: SPI MODE REQUIREMENTS (MASTER MODE)**

| Standard Operating Conditions (unless otherwise stated) |                                                |                                                    |                           |                   |                           |       |                                   |
|---------------------------------------------------------|------------------------------------------------|----------------------------------------------------|---------------------------|-------------------|---------------------------|-------|-----------------------------------|
| Param No.                                               | Symbol                                         | Characteristic                                     | Min.                      | Typ†              | Max.                      | Units | Conditions                        |
|                                                         | T <sub>SCK</sub>                               | SCK Cycle Time (2x Prescaled)                      | 61                        | —                 | —                         | ns    | Transmit only mode                |
|                                                         |                                                |                                                    | —                         | 16 <sup>(1)</sup> | —                         | MHz   |                                   |
|                                                         |                                                |                                                    | 95                        | —                 | —                         | ns    | Full-duplex mode                  |
|                                                         |                                                |                                                    | —                         | 10 <sup>(1)</sup> | —                         | MHz   |                                   |
| SP70*                                                   | T <sub>ssL2scH</sub> ,<br>T <sub>ssL2scL</sub> | SDO to SCK↓ or SCK↑ input                          | T <sub>SCK</sub>          | —                 | —                         | ns    | FST = 0                           |
|                                                         |                                                |                                                    | 0                         | —                 | —                         | ns    | FST = 1                           |
| SP71*                                                   | T <sub>scH</sub>                               | SCK output high time                               | 0.5 T <sub>SCK</sub> - 12 | —                 | 0.5 T <sub>SCK</sub> + 12 | ns    |                                   |
| SP72*                                                   | T <sub>scL</sub>                               | SCK output low time                                | 0.5 T <sub>SCK</sub> - 12 | —                 | 0.5 T <sub>SCK</sub> + 12 | ns    |                                   |
| SP73*                                                   | T <sub>dIV2scH</sub> ,<br>T <sub>dIV2scL</sub> | Setup time of SDI data input to SCK edge           | 85                        | —                 | —                         | ns    |                                   |
| SP74*                                                   | T <sub>sch2dIL</sub> ,<br>T <sub>scL2dIL</sub> | Hold time of SDI data input to SCK edge            | 0                         | —                 | —                         | ns    |                                   |
|                                                         |                                                | Hold time of SDI data input to final SCK           | 0.5 T <sub>SCK</sub>      | —                 | —                         | ns    | CKE = 0, SMP = 1                  |
| SP75*                                                   | T <sub>doR</sub>                               | SDO data output rise time                          | —                         | 10                | 25                        | ns    | C <sub>L</sub> = 50 pF            |
| SP76*                                                   | T <sub>doF</sub>                               | SDO data output fall time                          | —                         | 10                | 25                        | ns    | C <sub>L</sub> = 50 pF            |
| SP78*                                                   | T <sub>scR</sub>                               | SCK output rise time                               | —                         | 10                | 25                        | ns    | C <sub>L</sub> = 50 pF            |
| SP79*                                                   | T <sub>scF</sub>                               | SCK output fall time                               | —                         | 10                | 25                        | ns    | C <sub>L</sub> = 50 pF            |
| SP80*                                                   | T <sub>sch2doV</sub> ,<br>T <sub>scL2doV</sub> | SDO data output valid after SCK edge               | - 15                      | —                 | 15                        | ns    | C <sub>L</sub> = 20 pF            |
| SP81*                                                   | T <sub>doV2scH</sub> ,<br>T <sub>doV2scL</sub> | SDO data output valid to first SCK edge            | T <sub>SCK</sub> - 10     | —                 | —                         | ns    | C <sub>L</sub> = 20 pF<br>CKE = 1 |
| SP82*                                                   | T <sub>ssL2doV</sub>                           | SDO data output valid after $\overline{SS}$ ↓ edge | —                         | —                 | 50                        | ns    | C <sub>L</sub> = 20 pF            |
| SP83*                                                   | T <sub>sch2ssH</sub> ,<br>T <sub>scL2ssH</sub> | $\overline{SS}$ ↑ after last SCK edge              | 0.5 T <sub>SCK</sub> - 10 | —                 | —                         | ns    |                                   |
| SP84*                                                   | T <sub>ssH2ssL</sub>                           | $\overline{SS}$ ↑ to $\overline{SS}$ ↓ edge        | 0.5 T <sub>SCK</sub> - 10 | —                 | —                         | ns    |                                   |

\* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** SPIxCON1.SMP bit must be set and the slew rate control must be disabled on the clock and data pins (clear the corresponding bits in SLRCONx register) for SPI to operate over 4 MHz.

**TABLE 45-22: SPI MODE REQUIREMENTS (SLAVE MODE)**

| Standard Operating Conditions (unless otherwise stated) |                                                |                                                                       |      |                   |      |       |                        |
|---------------------------------------------------------|------------------------------------------------|-----------------------------------------------------------------------|------|-------------------|------|-------|------------------------|
| Param No.                                               | Symbol                                         | Characteristic                                                        | Min. | Typ†              | Max. | Units | Conditions             |
|                                                         | T <sub>SCK</sub>                               | SCK Total Cycle Time                                                  | 47   | —                 | —    | ns    | Receive only mode      |
|                                                         |                                                |                                                                       | —    | 20 <sup>(1)</sup> | —    | MHz   |                        |
|                                                         |                                                |                                                                       | 95   | —                 | —    | ns    | Full duplex mode       |
|                                                         |                                                |                                                                       | —    | 10 <sup>(1)</sup> | —    | MHz   |                        |
| SP70*                                                   | T <sub>ssL2scH</sub> ,<br>T <sub>ssL2scL</sub> | $\overline{SS}\downarrow$ to SCK $\downarrow$ or SCK $\uparrow$ input | 0    | —                 | —    | ns    | CKE = 0                |
|                                                         |                                                |                                                                       | 25   | —                 | —    | ns    | CKE = 1                |
| SP71*                                                   | T <sub>scH</sub>                               | SCK input high time                                                   | 20   | —                 | —    | ns    |                        |
| SP72*                                                   | T <sub>scL</sub>                               | SCK input low time                                                    | 20   | —                 | —    | ns    |                        |
| SP73*                                                   | T <sub>dIV2scH</sub> ,<br>T <sub>dIV2scL</sub> | Setup time of SDI data input to SCK edge                              | 10   | —                 | —    | ns    |                        |
| SP74*                                                   | T <sub>sch2diL</sub> ,<br>T <sub>scL2diL</sub> | Hold time of SDI data input to SCK edge                               | 0    | —                 | —    | ns    |                        |
| SP75*                                                   | T <sub>doR</sub>                               | SDO data output rise time                                             | —    | 10                | 25   | ns    | C <sub>L</sub> = 50 pF |
| SP76*                                                   | T <sub>doF</sub>                               | SDO data output fall time                                             | —    | 10                | 25   | ns    | C <sub>L</sub> = 50 pF |
| SP77*                                                   | T <sub>ssH2doZ</sub>                           | $\overline{SS}\uparrow$ to SDO output high-impedance                  | —    | —                 | 85   | ns    |                        |
| SP80*                                                   | T <sub>sch2doV</sub> ,<br>T <sub>scL2doV</sub> | SDO data output valid after SCK edge                                  | —    | —                 | 85   | ns    |                        |
| SP82*                                                   | T <sub>ssL2doV</sub>                           | SDO data output valid after $\overline{SS}\downarrow$ edge            | —    | —                 | 85   | ns    |                        |
| SP83*                                                   | T <sub>sch2ssH</sub> ,<br>T <sub>scL2ssH</sub> | $\overline{SS}\uparrow$ after SCK edge                                | 20   | —                 | —    | ns    |                        |
| SP84*                                                   | T <sub>ssH2ssL</sub>                           | $\overline{SS}\uparrow$ to $\overline{SS}\downarrow$ edge             | 47   | —                 | —    | ns    |                        |

\* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** SPIxCON1.SMP bit must be set and the slew rate control must be disabled on the clock and data pins (clear the corresponding bits in SLRCONx register) for SPI to operate over 4 MHz.

**FIGURE 45-18: I<sup>2</sup>C BUS START/STOP BITS TIMING**

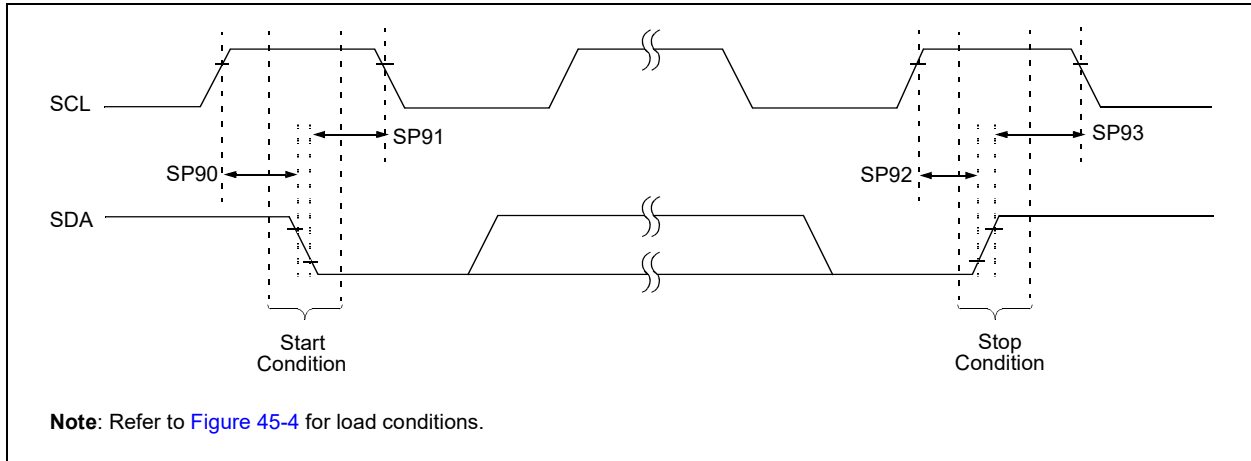


**TABLE 45-23: I<sup>2</sup>C BUS START/STOP BITS REQUIREMENTS**

| Standard Operating Conditions (unless otherwise stated) |         |                            |              |      |     |      |       |                                                       |
|---------------------------------------------------------|---------|----------------------------|--------------|------|-----|------|-------|-------------------------------------------------------|
| Param No.                                               | Symbol  | Characteristic             |              | Min. | Typ | Max. | Units | Conditions                                            |
| SP90*                                                   | TSU:STA | Start condition Setup time | 100 kHz mode | 4700 | —   | —    | ns    | Only relevant for Repeated Start condition            |
|                                                         |         |                            | 400 kHz mode | 600  | —   | —    |       |                                                       |
|                                                         |         |                            | 1 MHz mode   | 260  | —   | —    |       |                                                       |
| SP91*                                                   | THD:STA | Start condition Hold time  | 100 kHz mode | 4000 | —   | —    | ns    | After this period, the first clock pulse is generated |
|                                                         |         |                            | 400 kHz mode | 600  | —   | —    |       |                                                       |
|                                                         |         |                            | 1 MHz mode   | 260  | —   | —    |       |                                                       |
| SP92*                                                   | TSU:STO | Stop condition Setup time  | 100 kHz mode | 4000 | —   | —    | ns    |                                                       |
|                                                         |         |                            | 400 kHz mode | 600  | —   | —    |       |                                                       |
|                                                         |         |                            | 1 MHz mode   | 260  | —   | —    |       |                                                       |
| SP93                                                    | THD:STO | Stop condition Hold time   | 100 kHz mode | 4700 | —   | —    | ns    |                                                       |
|                                                         |         |                            | 400 kHz mode | 1300 | —   | —    |       |                                                       |
|                                                         |         |                            | 1 MHz mode   | 500  | —   | —    |       |                                                       |

\* These parameters are characterized but not tested.

**FIGURE 45-19: I<sup>2</sup>C BUS DATA TIMING**

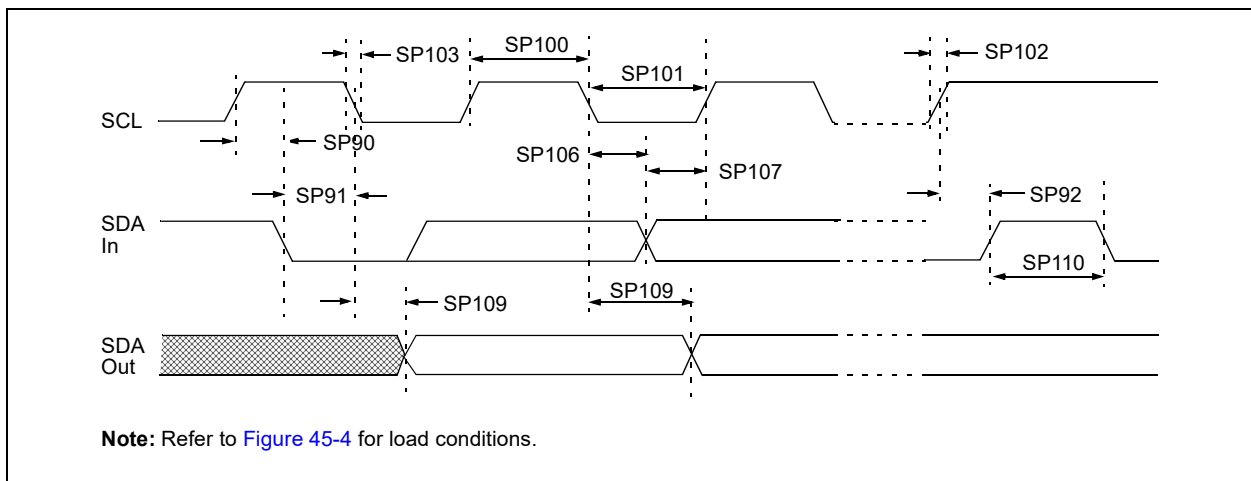


**TABLE 45-24: I<sup>2</sup>C BUS DATA REQUIREMENTS**

| Standard Operating Conditions (unless otherwise stated) |         |                         |              |                 |      |       |                                                               |
|---------------------------------------------------------|---------|-------------------------|--------------|-----------------|------|-------|---------------------------------------------------------------|
| Param. No.                                              | Symbol  | Characteristic          |              | Min.            | Max. | Units | Conditions                                                    |
| SP100*                                                  | THIGH   | Clock high time         | 100 kHz mode | 4000            | —    | ns    | Device must operate at a minimum of 1.5 MHz                   |
|                                                         |         |                         | 400 kHz mode | 600             | —    | ns    | Device must operate at a minimum of 10 MHz                    |
|                                                         |         |                         | 1 MHz mode   | 260             | —    | ns    | Device must operate at a minimum of 10 MHz                    |
| SP101*                                                  | TLOW    | Clock low time          | 100 kHz mode | 4700            | —    | ns    | Device must operate at a minimum of 1.5 MHz                   |
|                                                         |         |                         | 400 kHz mode | 1300            | —    | ns    | Device must operate at a minimum of 10 MHz                    |
|                                                         |         |                         | 1 MHz mode   | 500             | —    | —     | Device must operate at a minimum of 10 MHz                    |
| SP102*                                                  | TR      | SDA and SCL rise time   | 100 kHz mode | —               | 1000 | ns    |                                                               |
|                                                         |         |                         | 400 kHz mode | 20              | 300  | ns    | CB is specified to be from 10-400 pF                          |
|                                                         |         |                         | 1 MHz mode   | —               | 120  | ns    |                                                               |
| SP103*                                                  | TF      | SDA and SCL fall time   | 100 kHz mode | —               | 250  | ns    |                                                               |
|                                                         |         |                         | 400 kHz mode | 20 X (VDD/5.5V) | 250  | ns    | CB is specified to be from 10-400 pF                          |
|                                                         |         |                         | 1 MHz mode   | 20 X (VDD/5.5V) | 120  | ns    |                                                               |
| SP106*                                                  | THD:DAT | Data input hold time    | 100 kHz mode | 0               | —    | ns    |                                                               |
|                                                         |         |                         | 400 kHz mode | 0               | —    | ns    |                                                               |
|                                                         |         |                         | 1 MHz mode   | 0               | —    | ns    |                                                               |
| SP107*                                                  | TSU:DAT | Data input setup time   | 100 kHz mode | 250             | —    | ns    | (2)                                                           |
|                                                         |         |                         | 400 kHz mode | 100             | —    | ns    |                                                               |
|                                                         |         |                         | 1 MHz mode   | 50              | —    | ns    |                                                               |
| SP109*                                                  | TAA     | Output valid from clock | 100 kHz mode | —               | 3450 | ns    | (1)                                                           |
|                                                         |         |                         | 400 kHz mode | —               | 900  | ns    |                                                               |
|                                                         |         |                         | 1 MHz mode   | —               | 450  | ns    |                                                               |
| SP110*                                                  | TBUF    | Bus free time           | 100 kHz mode | 4700            | —    | ns    | Time the bus must be free before a new transmission can start |
|                                                         |         |                         | 400 kHz mode | 1300            | —    | ns    |                                                               |
|                                                         |         |                         | 1 MHz mode   | 500             | —    | ns    |                                                               |
| SP111                                                   | CB      | Bus capacitive loading  |              | —               | 400  | pF    |                                                               |

\* These parameters are characterized but not tested.

- Note 1:** As a transmitter, the device must provide this internal minimum delay time to bridge the undefined region (min. 300 ns) of the falling edge of SCL to avoid unintended generation of Start or Stop conditions.
- 2:** A Fast mode (400 kHz) I<sup>2</sup>C bus device can be used in a Standard mode (100 kHz) I<sup>2</sup>C bus system, but the requirement TSU:DAT ≥ 250 ns must then be met. This will automatically be the case if the device does not stretch the low period of the SCL signal. If such a device does stretch the low period of the SCL signal, it must output the next data bit to the SDA line TR max. + TSU:DAT = 1000 + 250 = 1250 ns (according to the Standard mode I<sup>2</sup>C bus specification), before the SCL line is released.

**TABLE 45-25: TEMPERATURE INDICATOR REQUIREMENTS**

| Standard Operating Conditions (unless otherwise stated) |         |                                    |            |      |        |      |       |
|---------------------------------------------------------|---------|------------------------------------|------------|------|--------|------|-------|
| Param No.                                               | Symbol  | Characteristic                     |            | Min. | Typ†   | Max. | Units |
| TS01*                                                   | TACQMIN | Minimum ADC Acquisition Time Delay |            | —    | 25     | —    | μs    |
| TS02*                                                   | Mv      | Voltage Sensitivity                | High Range | —    | -3.684 | —    | mV/°C |
|                                                         |         |                                    | Low Range  | —    | -2.456 | —    | mV/°C |

\* These parameters are characterized but not tested.

† Data in “Typ” column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.



## 46.0 DC AND AC CHARACTERISTICS GRAPHS AND CHARTS

The graphs and tables provided in this section are for **design guidance** and are **not tested**.

In some graphs or tables, the data presented are **outside specified operating range** (i.e., outside specified  $V_{DD}$  range). This is for **information only** and devices are ensured to operate properly only within the specified range.

Unless otherwise noted, all graphs apply to both the L and LF devices.

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Note:</b> | The graphs and tables provided following this note are a statistical summary based on a limited number of samples and are provided for informational purposes only. The performance characteristics listed herein are not tested or guaranteed. In some graphs or tables, the data presented may be outside the specified operating range (e.g., outside specified power supply range) and therefore, outside the warranted range. |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**“Typical” represents the mean of the distribution at 25°C. “Maximum”, “Max.”, “Minimum” or “Min.” represents  $(\text{mean} + 3\sigma)$  or  $(\text{mean} - 3\sigma)$  respectively, where  $\sigma$  is a standard deviation, over each temperature range.**

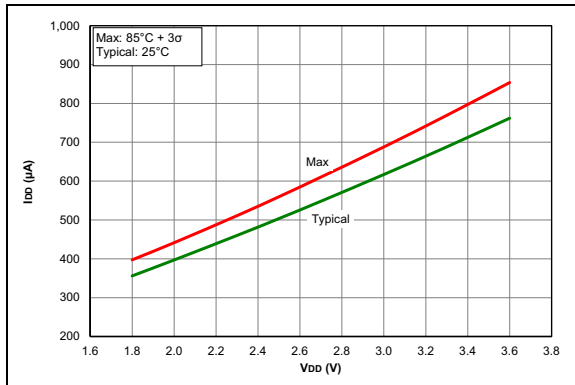


**FIGURE 46-1:**  $I_{DD}$ , XT Oscillator, 4 MHz, PIC18LF25/26K83 Only.

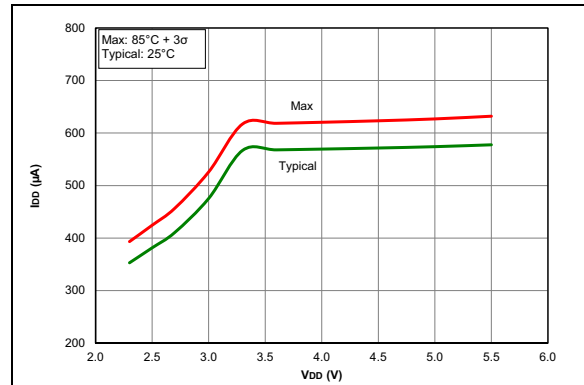


**FIGURE 46-4:**  $I_{DD}$ , XT Oscillator, 4 MHz, PMD's All '1's, PIC18F25/26K83 Only.

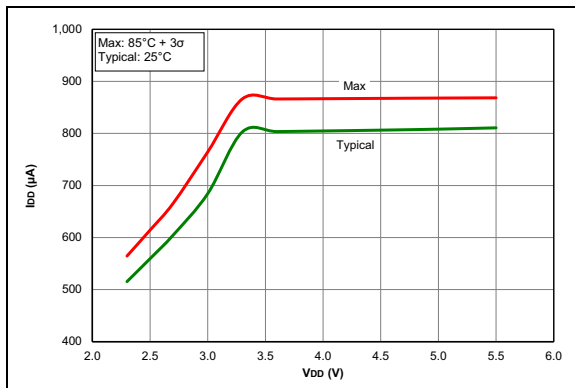


**FIGURE 46-2:**  $I_{DD}$ , XT Oscillator, 4 MHz, PIC18F25/26K83 Only.

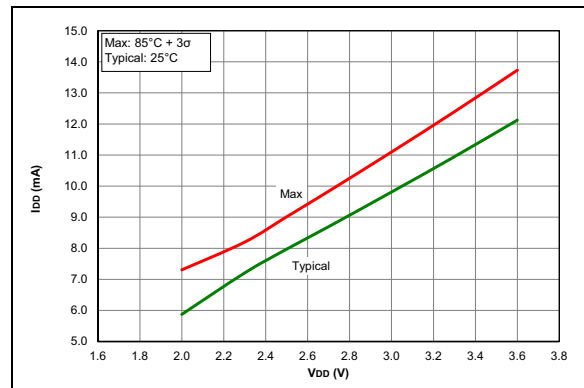


**FIGURE 46-5:**  $I_{DD}$ , HS+PLL Oscillator, 64 MHz, PIC18LF25/26K83 Only.

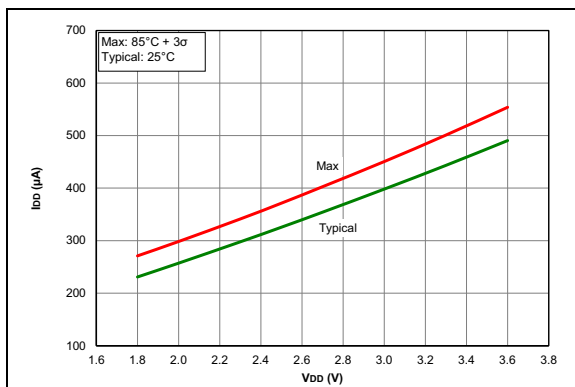


**FIGURE 46-3:**  $I_{DD}$ , XT Oscillator, 4 MHz, PMD's All '1's, PIC18LF25/26K83 Only.

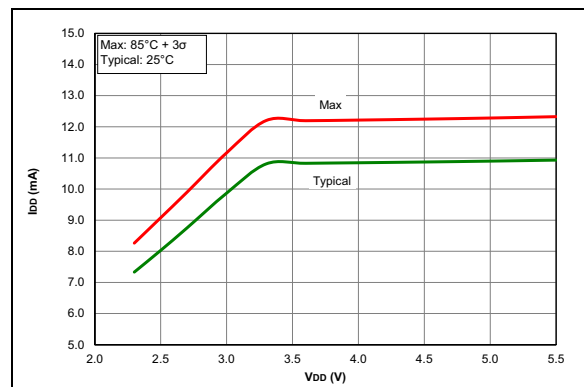


**FIGURE 46-6:**  $I_{DD}$ , HS+PLL Oscillator, 64 MHz, PIC18F25/26K83 Only.

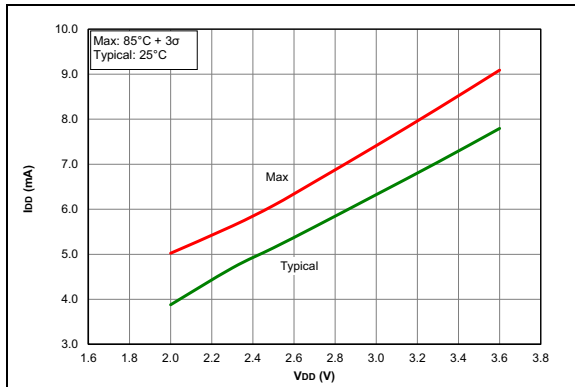


**FIGURE 46-7:**  $I_{DD}$ , HS+PLL Oscillator, 64 MHz, PMD's All '1's, PIC18LF25/26K83 Only.

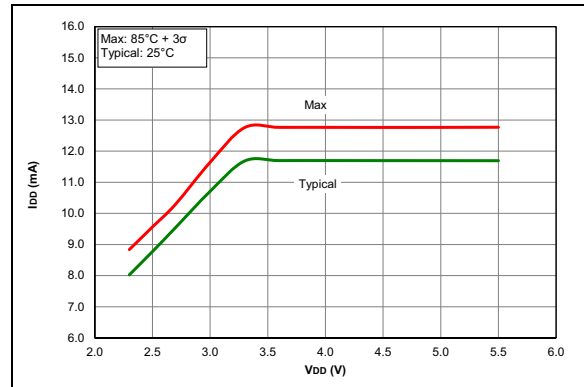


**FIGURE 46-10:**  $I_{DD}$ , HFINTOSC Mode,  $F_{osc} = 64$  MHz, PIC18F25/26K83 Only.

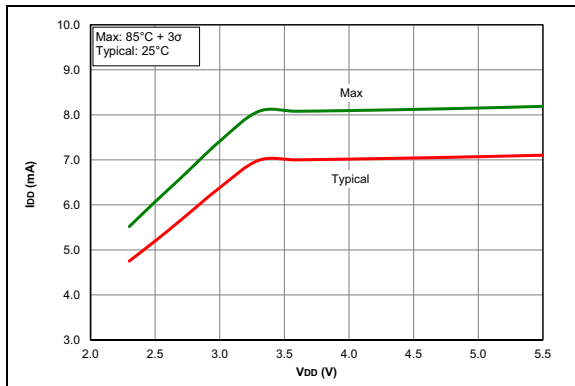


**FIGURE 46-8:**  $I_{DD}$ , HS+PLL Oscillator, 64 MHz, PMD's All '1's, PIC18F25/26K83 Only.

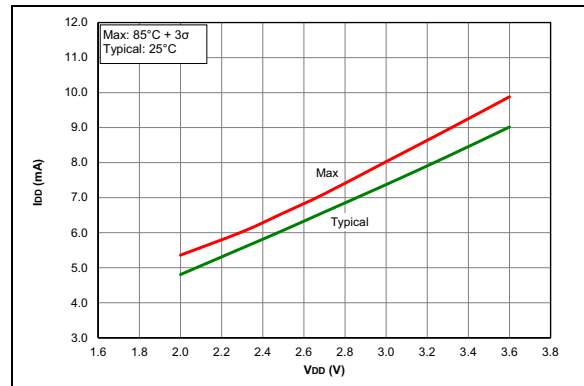


**FIGURE 46-11:**  $I_{DD}$ , HFINTOSC Mode,  $F_{osc} = 64$  MHz, PMD's All '1's, PIC18LF25/26K83 Only.

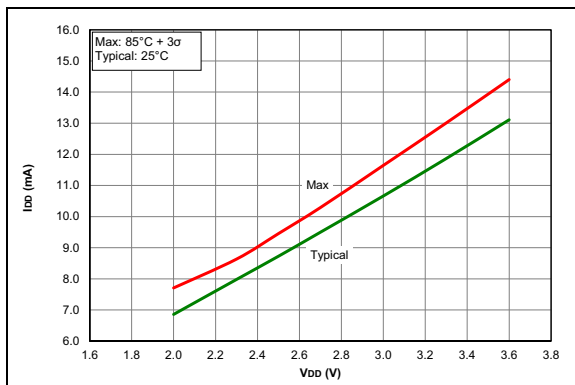


**FIGURE 46-9:**  $I_{DD}$ , HFINTOSC Mode,  $F_{osc} = 64$  MHz, PIC18LF25/26K83 Only.

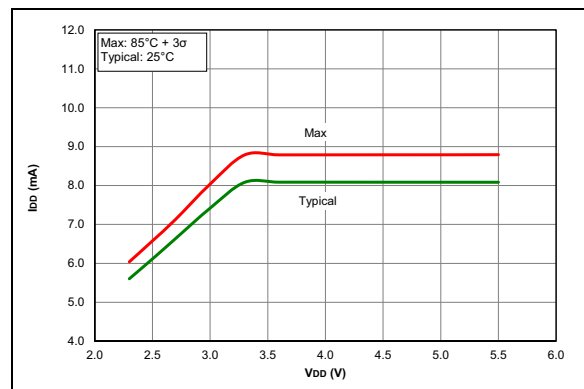


**FIGURE 46-12:**  $I_{DD}$ , HFINTOSC Mode,  $F_{osc} = 16$  MHz, PIC18F25/26K83 Only.

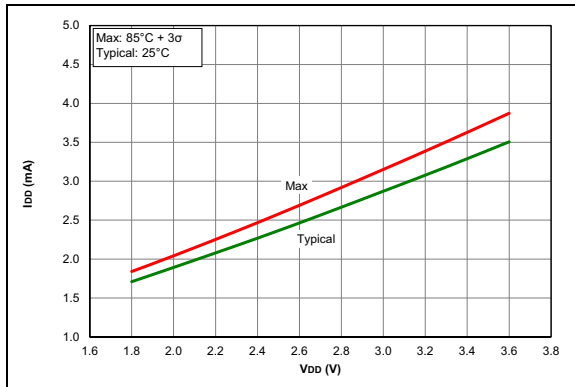


**FIGURE 46-13:**  $I_{DD}$ , HFINTOSC Mode,  $F_{osc} = 16$  MHz, PIC18LF25/26K83 Only.

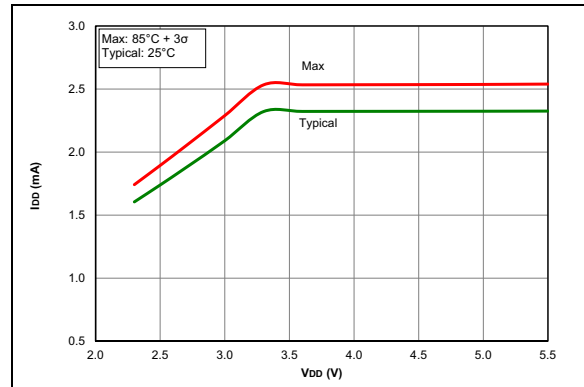


**FIGURE 46-16:**  $I_{DD}$ , HFINTOSC Mode,  $F_{osc} = 16$  MHz, PMD's All '1's, PIC18F25/26K83 Only.

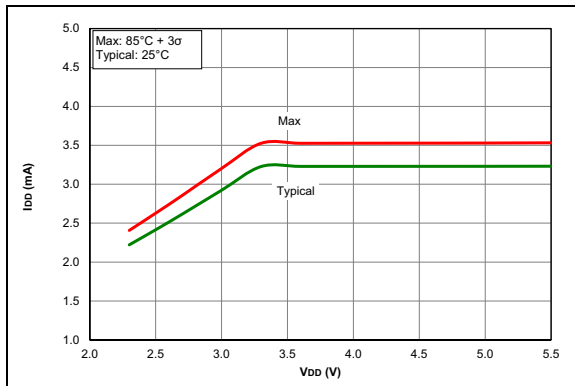


**FIGURE 46-14:**  $I_{DD}$ , HFINTOSC Mode,  $F_{osc} = 16$  MHz, PIC18F25/26K83 Only.

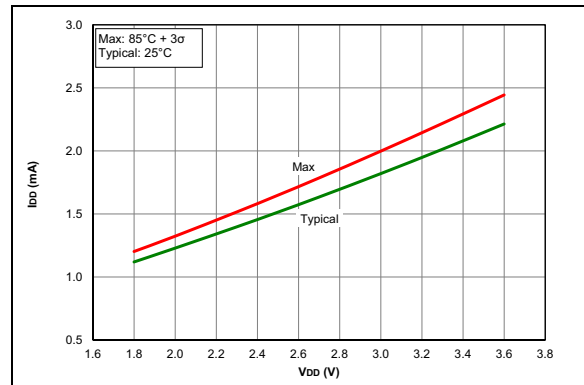


**FIGURE 46-17:**  $I_{DD}$ , HFINTOSC Idle Mode,  $F_{osc} = 16$  MHz, PIC18LF25/26K83 Only.

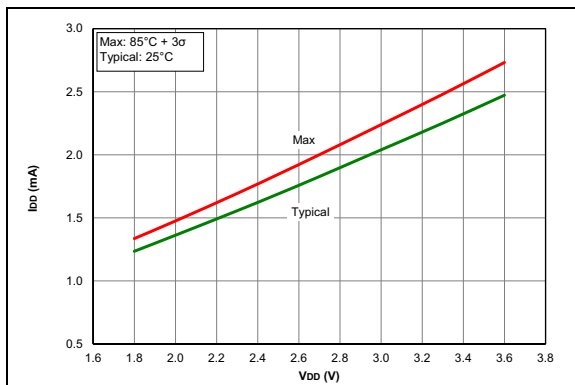


**FIGURE 46-15:**  $I_{DD}$ , HFINTOSC Mode,  $F_{osc} = 16$  MHz, PMD's All '1's, PIC18LF25/26K83 Only.

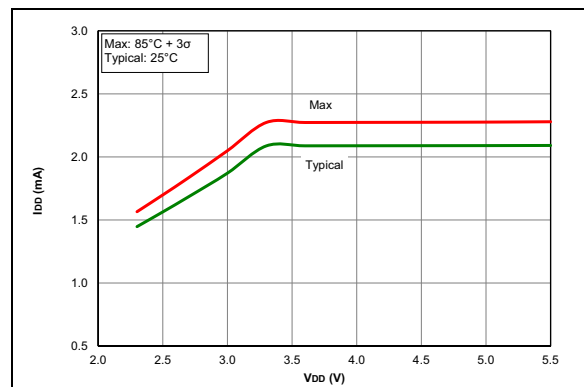


**FIGURE 46-18:**  $I_{DD}$ , HFINTOSC Idle Mode,  $F_{osc} = 16$  MHz, PIC18F25/26K83 Only.

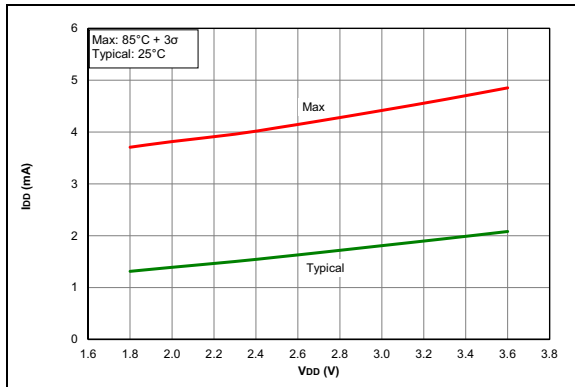


**FIGURE 46-19:**  $I_{DD}$ , HFINTOSC Doze Mode,  $F_{osc} = 16$  MHz, PIC18LF25/26K83 Only.

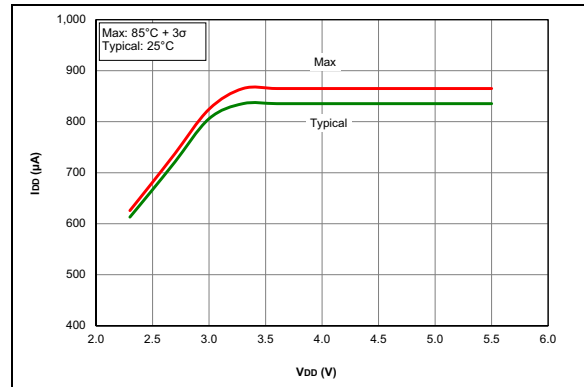


**FIGURE 46-22:**  $I_{DD}$ , XT Oscillator 4 MHz, PIC18F25/26K83 Only

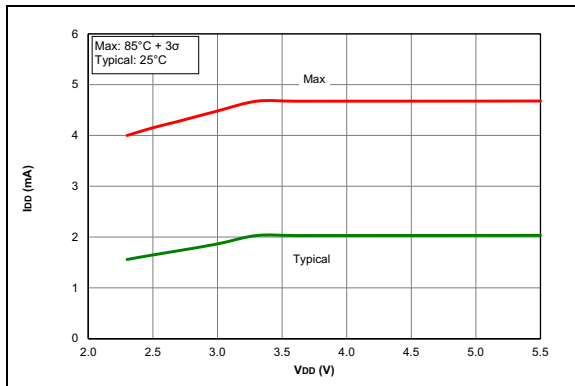


**FIGURE 46-20:**  $I_{DD}$ , HFINTOSC Doze Mode,  $F_{osc} = 16$  MHz, PIC18F25/26K83 Only.

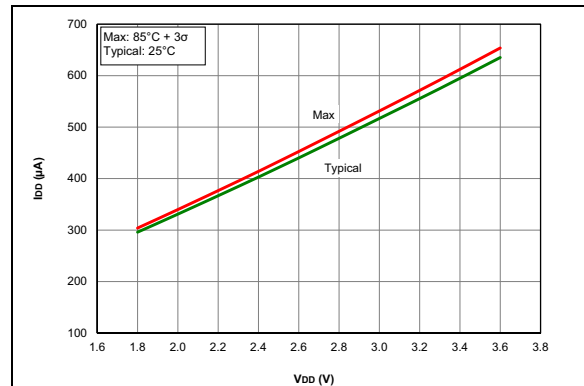


**FIGURE 46-23:**  $I_{DD}$ , XT Oscillator 4 MHz, PMD's All '1's, PIC18LF25/26K83 Only

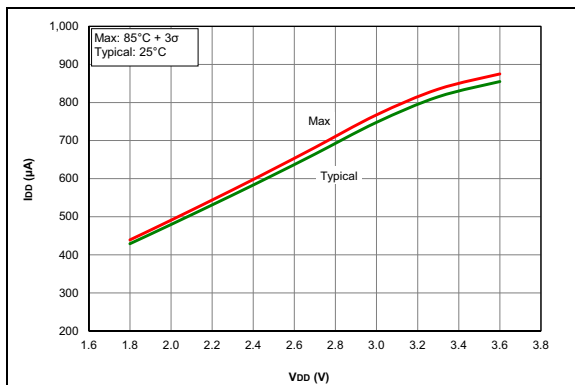


**FIGURE 46-21:**  $I_{DD}$ , XT Oscillator 4 MHz, PIC18LF25/26K83 Only

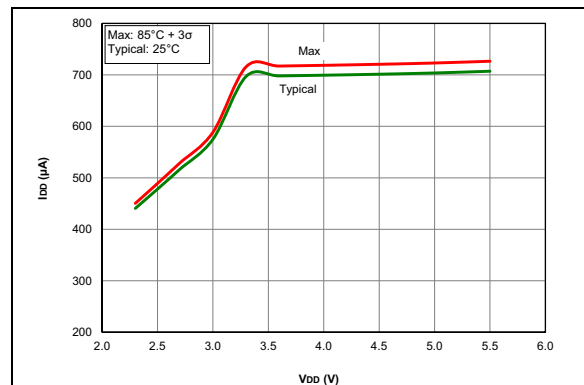


**FIGURE 46-24:**  $I_{DD}$ , XT Oscillator 4 MHz, PMD's All '1's, PIC18F25/26K83 Only

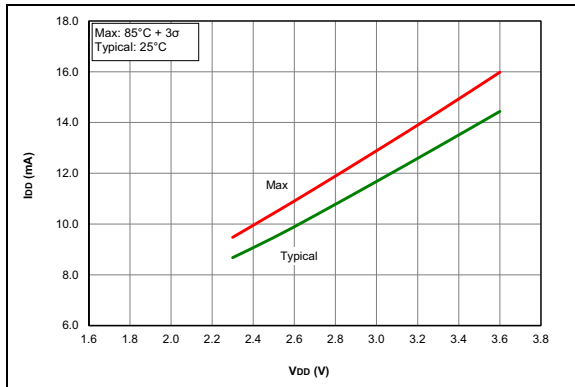


**FIGURE 46-25:**  $I_{DD}$ , HS+PLL Oscillator, 64 MHz, PIC18LF25/26K83 Only.

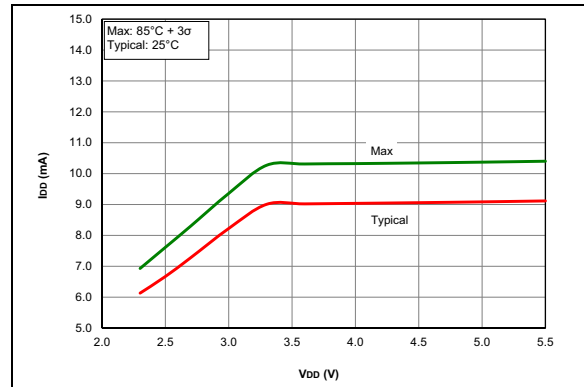


**FIGURE 46-28:**  $I_{DD}$ , HS+PLL Oscillator, 64 MHz, PMD's All '1's, PIC18F25/26K83 Only.

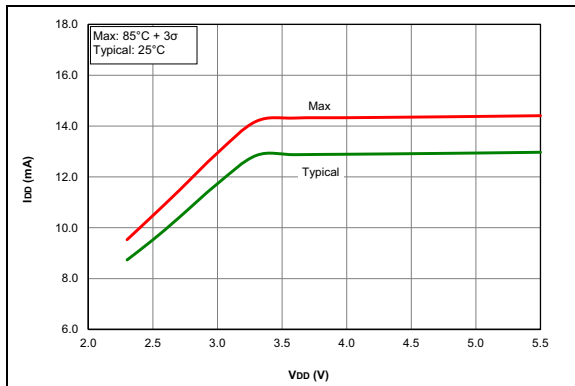


**FIGURE 46-26:**  $I_{DD}$ , HS+PLL Oscillator, 64 MHz, PIC18F25/26K83 Only.

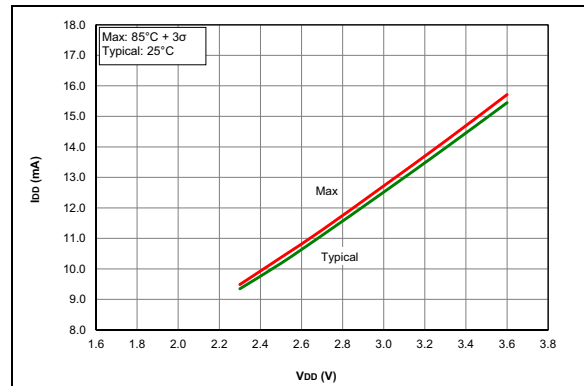


**FIGURE 46-29:**  $I_{DD}$ , HFINTOSC Mode, FOSC = 64 MHz, PIC18LF25/26K83 Only.

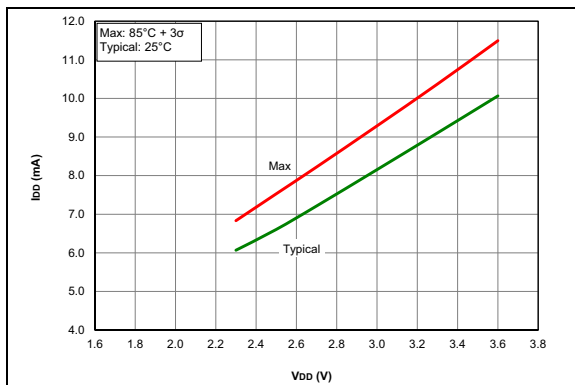


**FIGURE 46-27:**  $I_{DD}$ , HS+PLL Oscillator, 64 MHz, PMD's All '1's, PIC18LF25/26K83 Only.

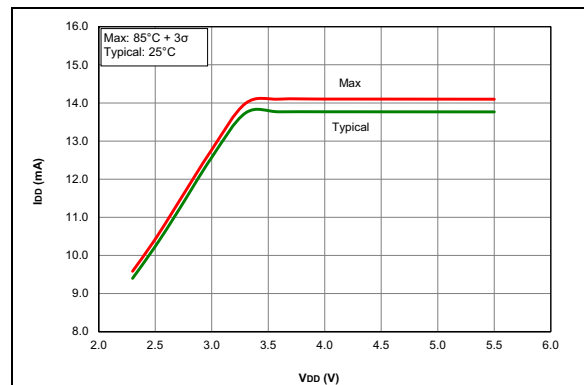


**FIGURE 46-30:**  $I_{DD}$ , HFINTOSC Mode, FOSC = 64 MHz, PIC18F25/26K83 Only.

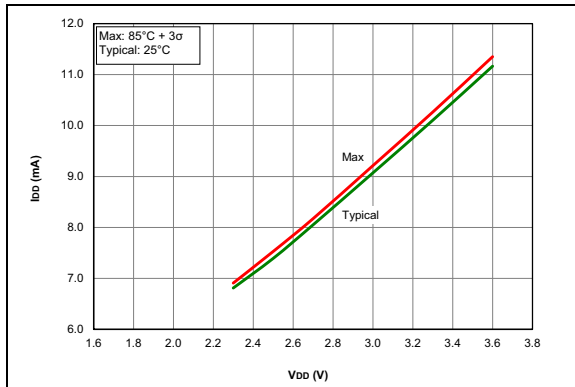


**FIGURE 46-31:**  $I_{DD}$ , HFINTOSC Mode,  $F_{osc} = 64$  MHz, PMD's All '1's, PIC18LF25/26K83 Only.

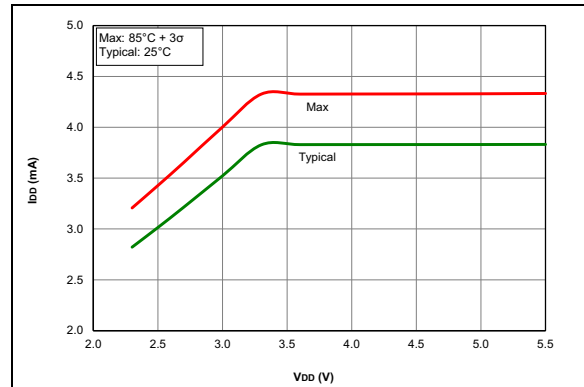


**FIGURE 46-34:**  $I_{DD}$ , HFINTOSC Mode,  $F_{osc} = 16$  MHz, PIC18F25/26K83 Only.

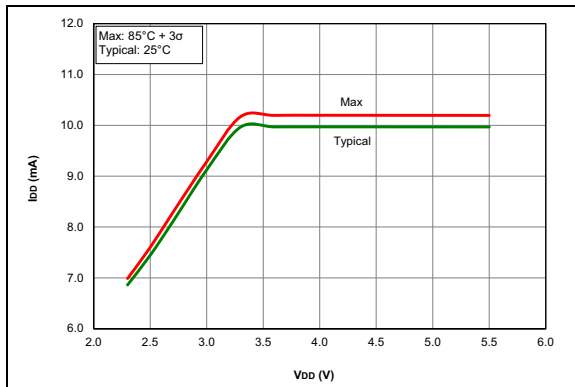


**FIGURE 46-32:**  $I_{DD}$ , HFINTOSC Mode,  $F_{osc} = 64$  MHz, PMD's All '1's, PIC18F25/26K83 Only.

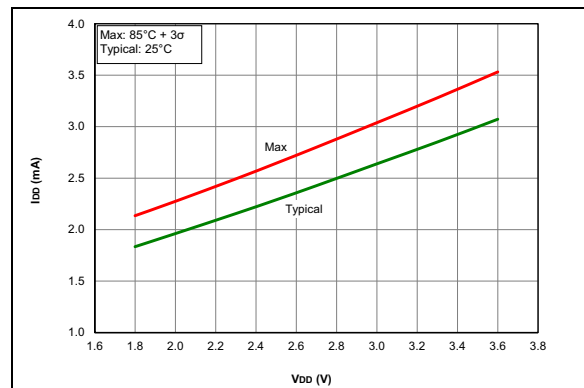


**FIGURE 46-35:**  $I_{DD}$ , HFINTOSC Mode,  $F_{osc} = 16$  MHz, PMD's All '1's, PIC18LF25/26K83 Only.

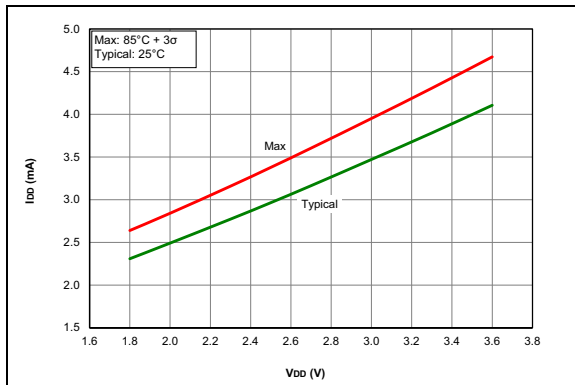


**FIGURE 46-33:**  $I_{DD}$ , HFINTOSC Mode,  $F_{osc} = 16$  MHz, PIC18LF25/26K83 Only.

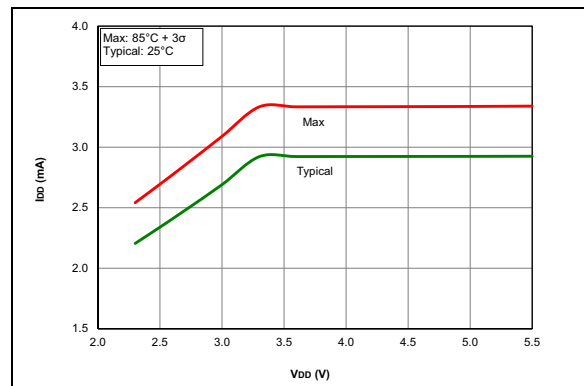


**FIGURE 46-36:**  $I_{DD}$ , HFINTOSC Mode,  $F_{osc} = 16$  MHz, PMD's All '1's, PIC18F25/26K83 Only.

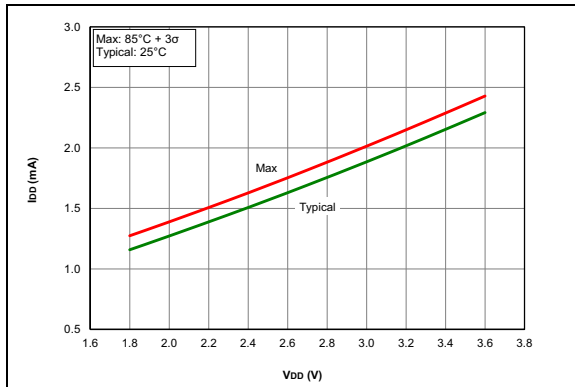


**FIGURE 46-37:**  $I_{DD}$ , HFINTOSC Idle Mode,  $F_{osc} = 16$  MHz, PIC18LF25/26K83 Only.

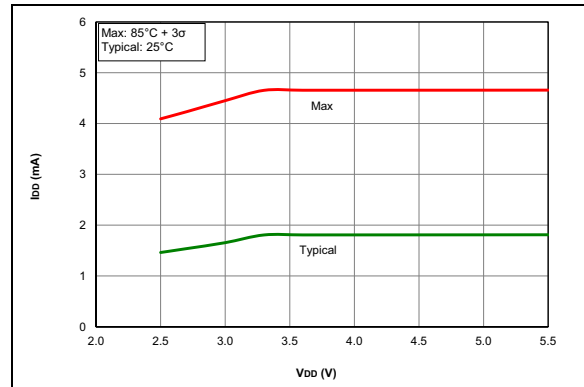


**FIGURE 46-40:**  $I_{DD}$ , HFINTOSC Doze Mode,  $F_{osc} = 16$  MHz, PIC18F25/26K83 Only.

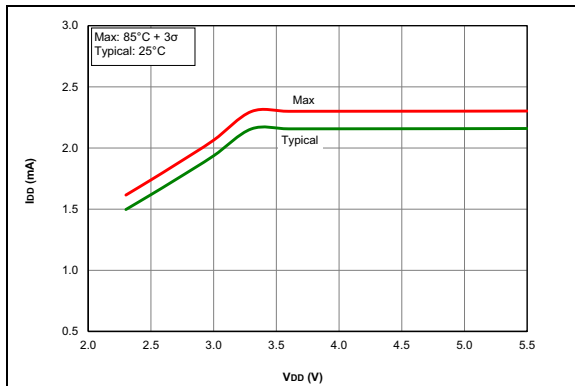


**FIGURE 46-38:**  $I_{DD}$ , HFINTOSC Idle Mode,  $F_{osc} = 16$  MHz, PIC18F25/26K83 Only.

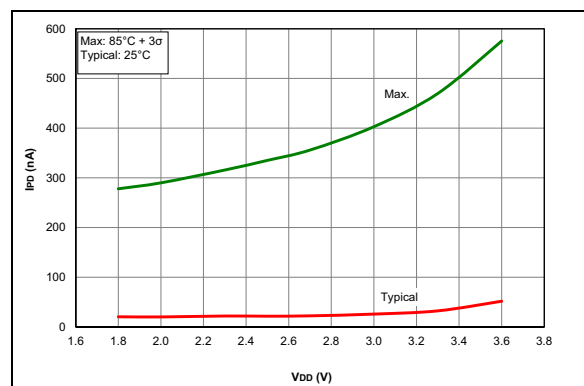


**FIGURE 46-41:**  $I_{PD}$ , Base, LP Sleep Mode, PIC18LF25/26K83 Only.

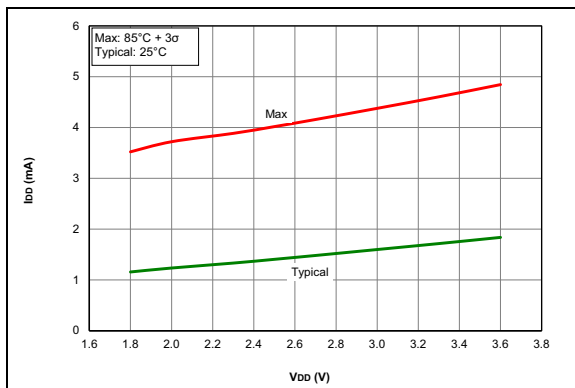


**FIGURE 46-39:**  $I_{DD}$ , HFINTOSC Doze Mode,  $F_{osc} = 16$  MHz, PIC18LF25/26K83 Only.

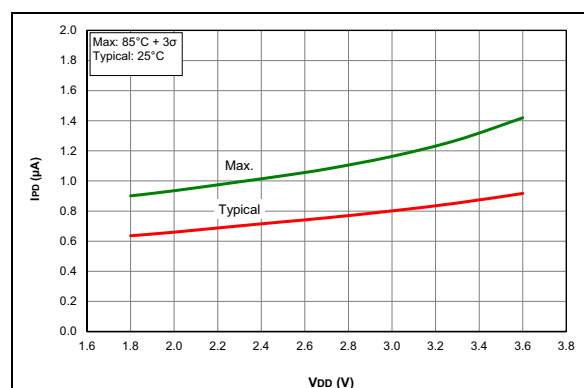


**FIGURE 46-42:**  $I_{PD}$ , Watchdog Timer (WDT), PIC18LF25/26K83 Only.



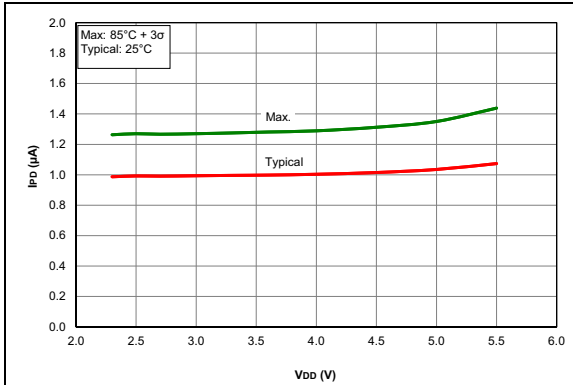


**FIGURE 46-43:** *I<sub>PO</sub>, Watchdog Timer (WDT), PIC18F25/26K83 Only.*

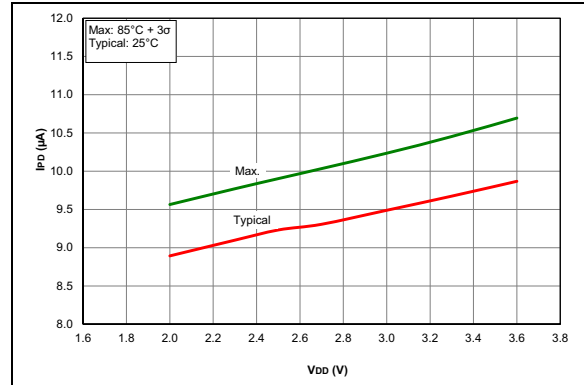


**FIGURE 46-46:** *I<sub>PO</sub>, Brown-Out Reset (BOR), PIC18LF25/26K83 Only.*

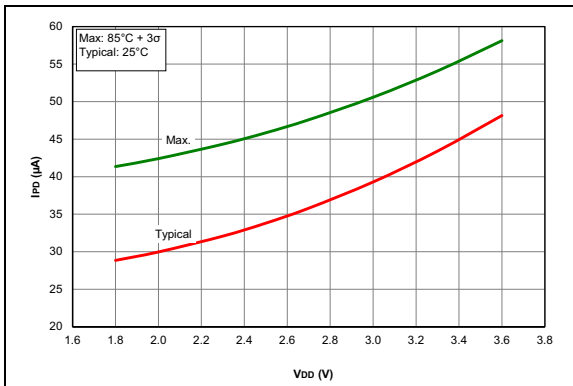


**FIGURE 46-44:** *I<sub>PO</sub>, Fixed Voltage Reference (FVR), PIC18LF25/26K83 Only.*

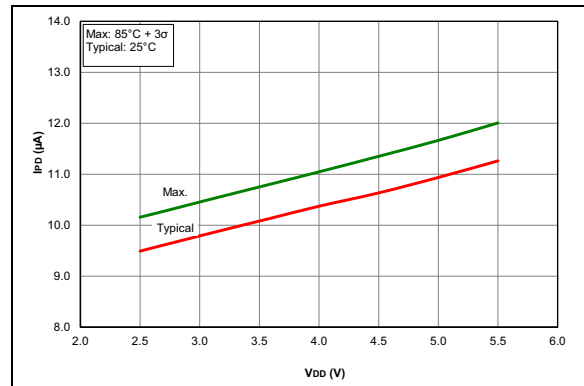


**FIGURE 46-47:** *I<sub>PO</sub>, Brown-Out Reset (BOR), PIC18F25/26K83 Only.*

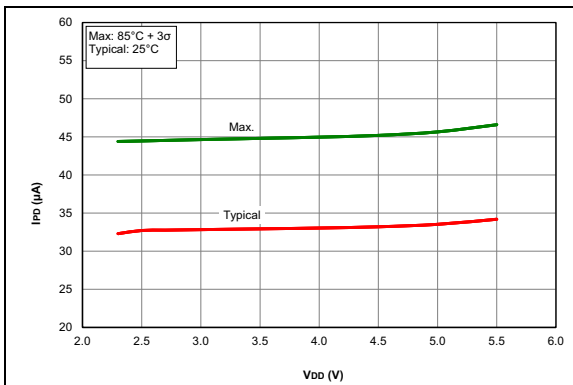


**FIGURE 46-45:** *I<sub>PO</sub>, Fixed Voltage Reference (FVR), PIC18F25/26K83 Only.*

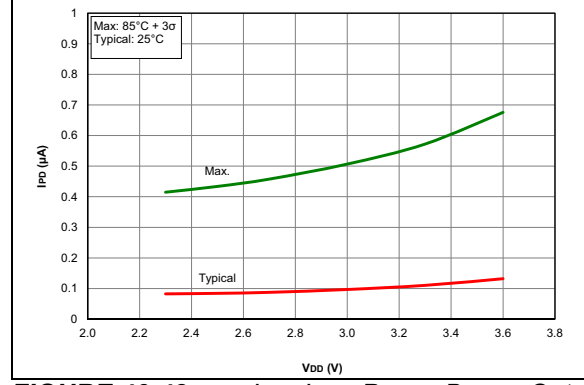


**FIGURE 46-48:** *I<sub>PO</sub>, Low-Power Brown-Out Reset (LPBOR), PIC18LF25/26K83 Only.*

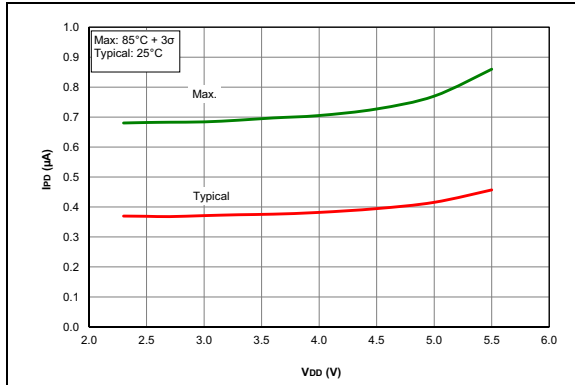


**FIGURE 46-49:** *IPD, Low-Power Brown-Out Reset (LPBOR), PIC18F25/26K83 Only.*

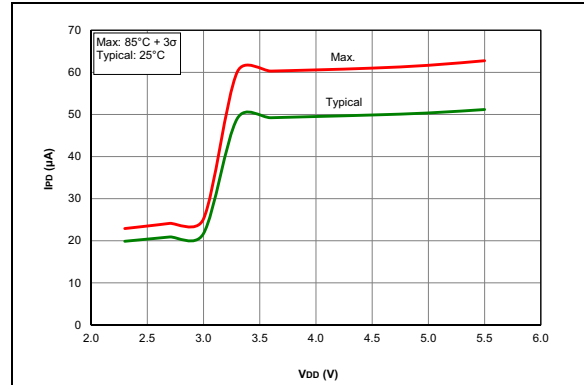


**FIGURE 46-52:** *IPD Base, NP Sleep Mode, PIC18F25/26K83 Only.*

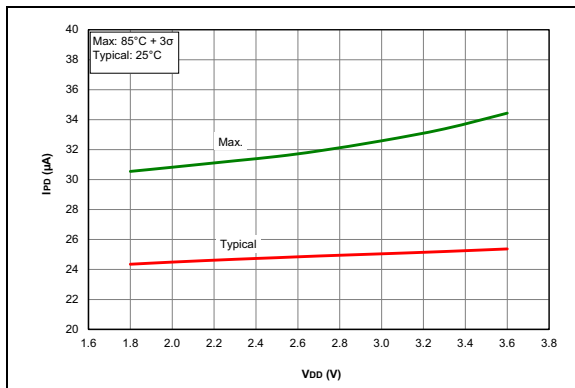


**FIGURE 46-50:** *IPD, Comparator, PIC18LF25/26K83 Only.*

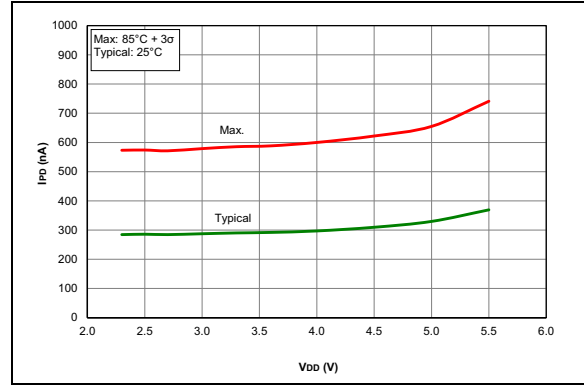


**FIGURE 46-53:** *IPD Base, LP Sleep Mode, PIC18F25/26K83 Only*

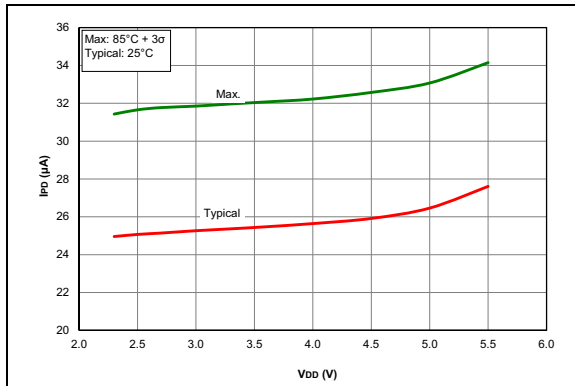


**FIGURE 46-51:** *IPD, Comparator, PIC18F25/26K83 Only.*

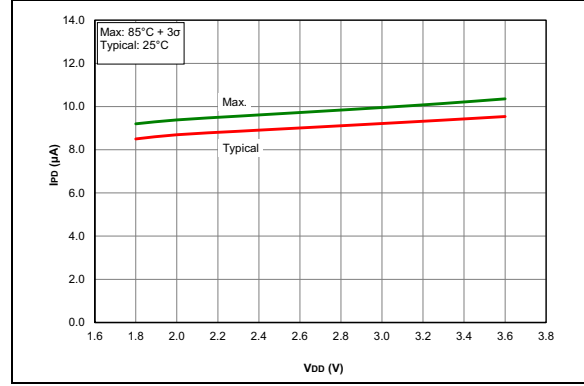


**FIGURE 46-54:** *IPD, High/Low Voltage detect (HLVD), PIC18LF25/26K83 Only.*

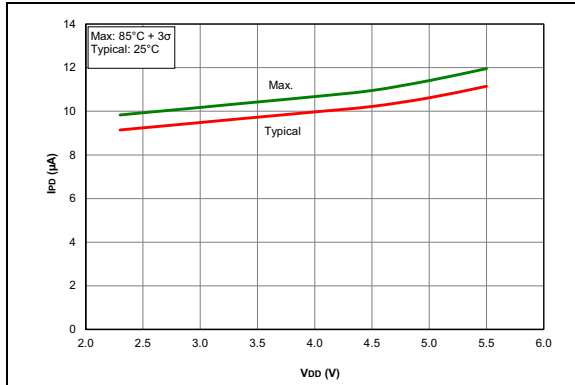


**FIGURE 46-55:** *IPD, High/Low Voltage detect (HLVD), PIC18F25/26K83 Only.*

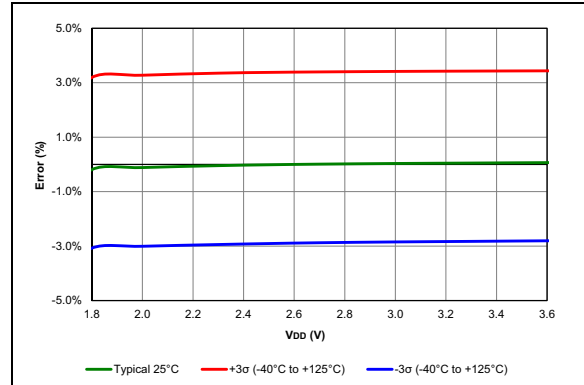


**FIGURE 46-58:** *Calibrated HFINTOSC, Typical Frequency Error, PIC18LF25/26K83 only.*

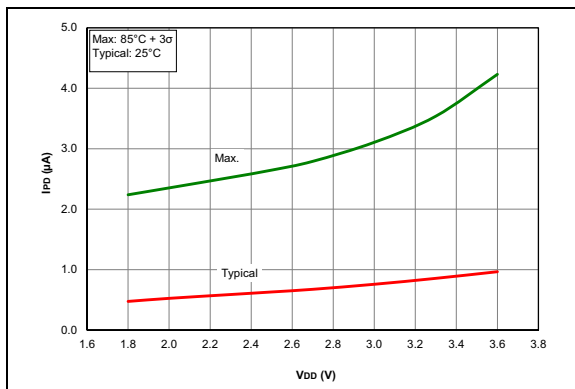


**FIGURE 46-56:** *IPD, Secondary Oscillator (SOSC), PIC18LF25/26K83 Only.*

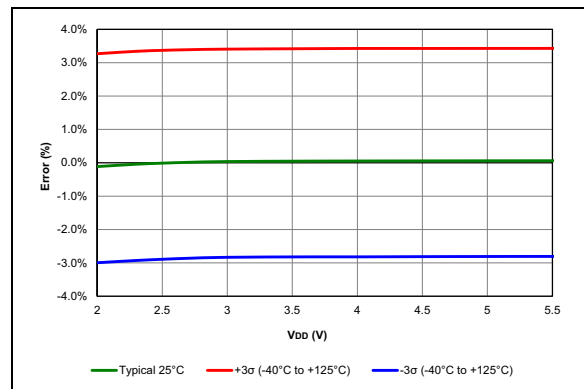


**FIGURE 46-59:** *Calibrated HFINTOSC, Typical Frequency Error, PIC18F25/26K83 only.*

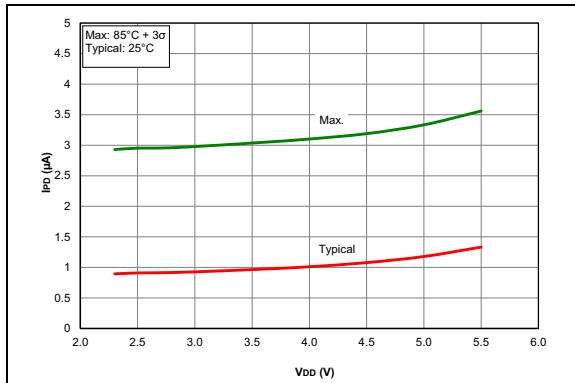


**FIGURE 46-57:** *IPD, Secondary Oscillator (SOSC), PIC18F25/26K83 Only.*

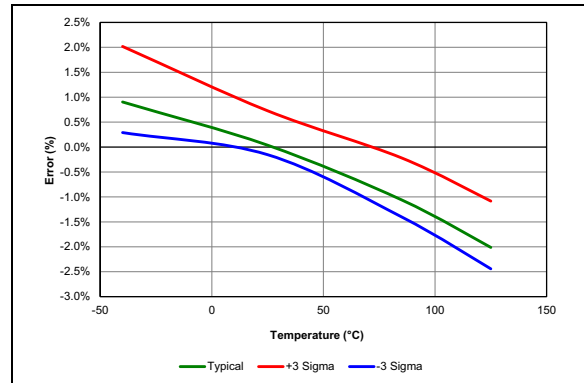


**FIGURE 46-60:** *HFINTOSC Frequency Error, VDD = 3.0V.*

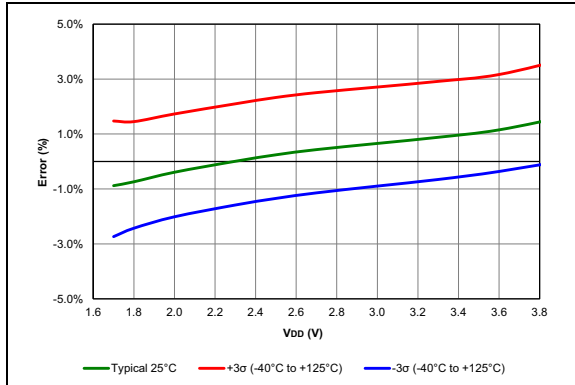


**FIGURE 46-61:** LFINTOSC Typical Frequency Error, PIC18LF25/26K83 Only.

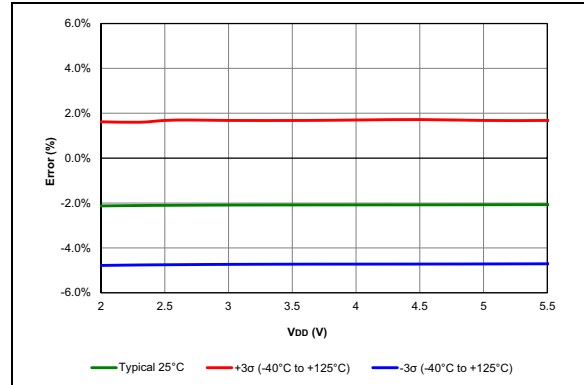


**FIGURE 46-64:** Low-Power Optimized HFINTOSC Typical Frequency Error, PIC18F25/26K83 Only.

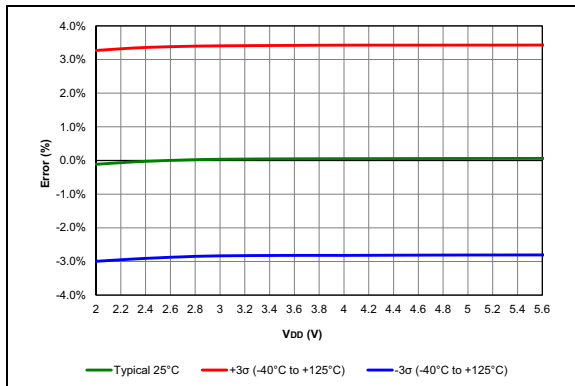


**FIGURE 46-62:** LFINTOSC Typical Frequency Error, PIC18F25/26K83 Only.

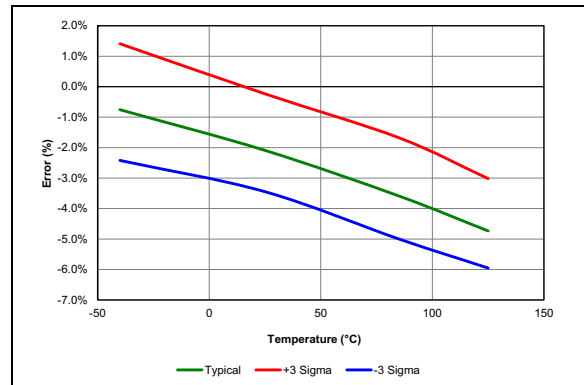


**FIGURE 46-65:** Low-Power Optimized HFINTOSC Frequency Error,  $V_{DD} = 3.0V$ .

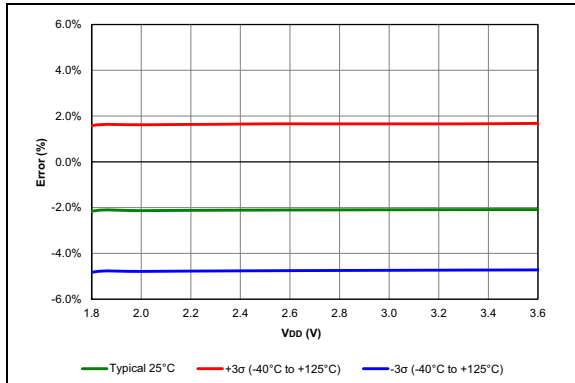


**FIGURE 46-63:** Low-Power Optimized HFINTOSC Typical Frequency Error, PIC18LF25/26K83 Only.

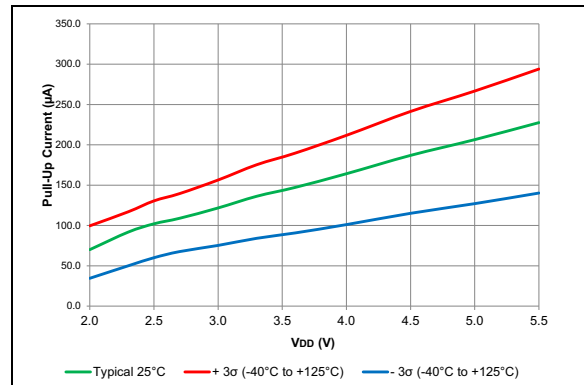


**FIGURE 46-66:** Weak Pull-Up Current, PIC18F25/26K83 Only.

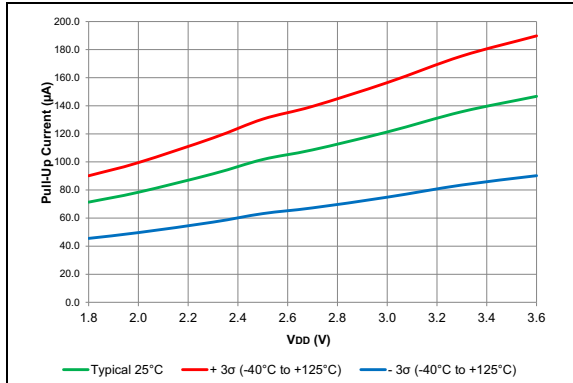


**FIGURE 46-67:** Weak Pull-Up Current, PIC18LF25/26K83 Only.

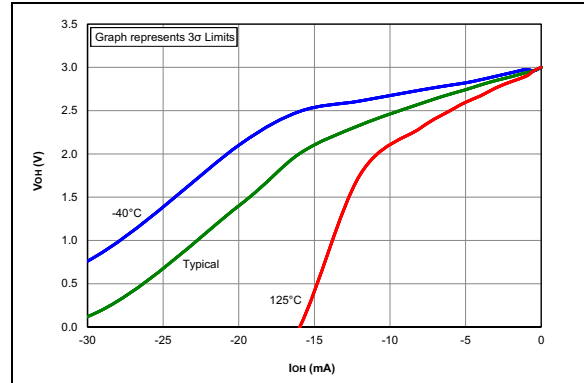


**FIGURE 46-70:**  $V_{OH}$  vs.  $I_{OH}$  Over Temperature,  $V_{DD} = 3.0V$ .

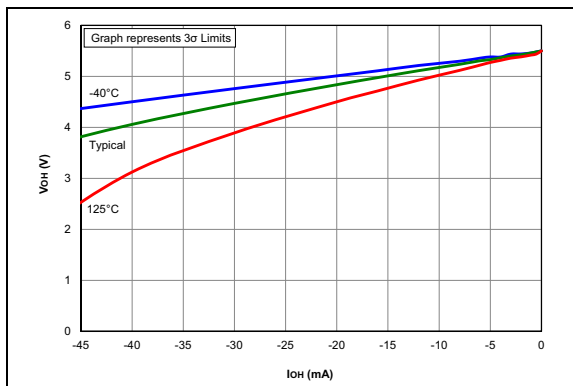


**FIGURE 46-68:**  $V_{OH}$  vs.  $I_{OH}$  Over Temperature,  $V_{DD} = 5.5V$ , PIC18F25/26K83 Only.

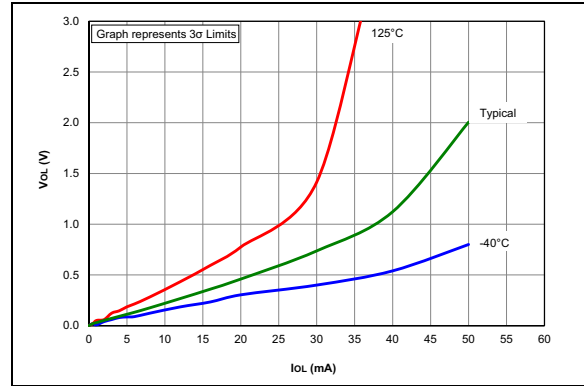


**FIGURE 46-71:**  $V_{OL}$  vs.  $I_{OL}$  Over Temperature,  $V_{DD} = 3.0V$ .

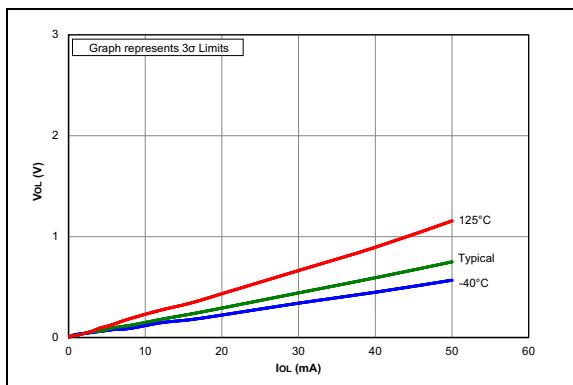


**FIGURE 46-69:**  $V_{OL}$  vs.  $I_{OL}$  Over Temperature,  $V_{DD} = 5.5V$ , PIC18F25/26K83 Only.

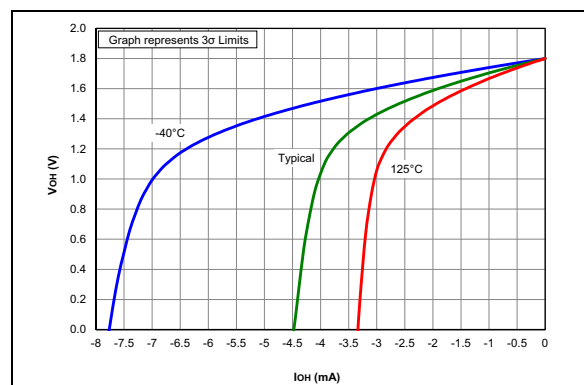


**FIGURE 46-72:**  $V_{OH}$  vs.  $I_{OH}$  Over Temperature,  $V_{DD} = 1.8V$ , PIC18LF25/26K83 Only.

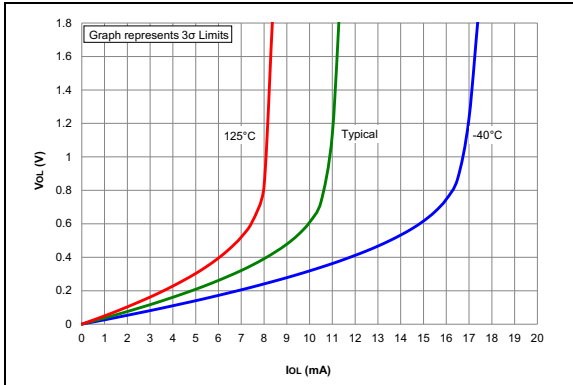


**FIGURE 46-73:** VOL vs. IOL Over Temperature,  $V_{DD} = 1.8V$ , PIC18LF25/26K83 Only

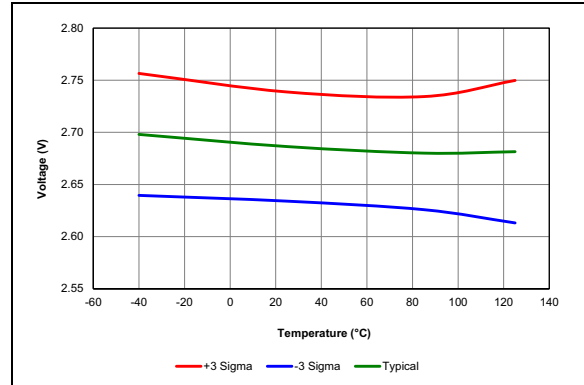


**FIGURE 46-76:** Brown-Out Reset Voltage, Trip Point (BORV = 01).

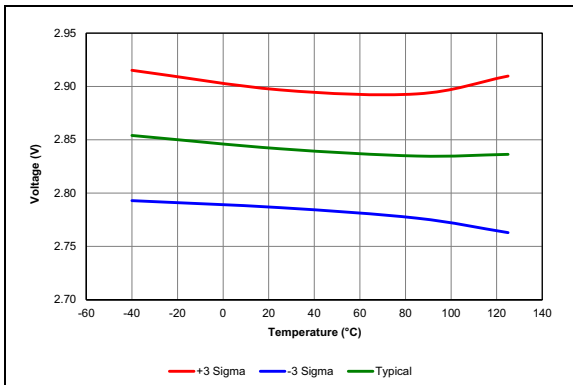


**FIGURE 46-74:** Brown-Out Reset Voltage, Trip Point (BORV = 00).

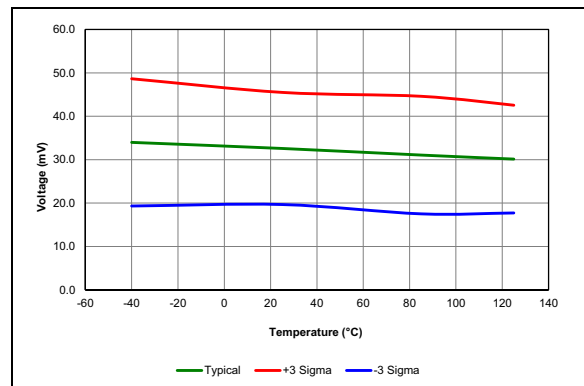


**FIGURE 46-77:** Brown-Out Reset Hysteresis, Trip Point (BORV = 01).

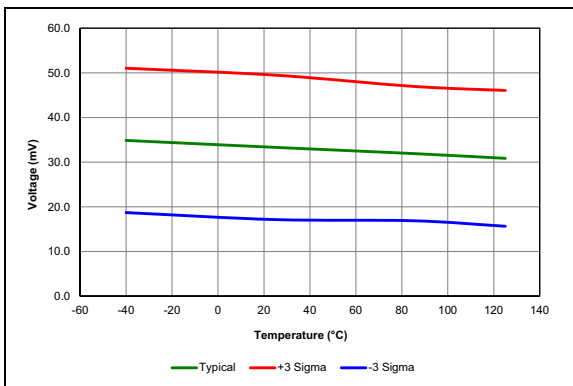


**FIGURE 46-75:** Brown-Out Reset Hysteresis, Trip Point (BORV = 00).

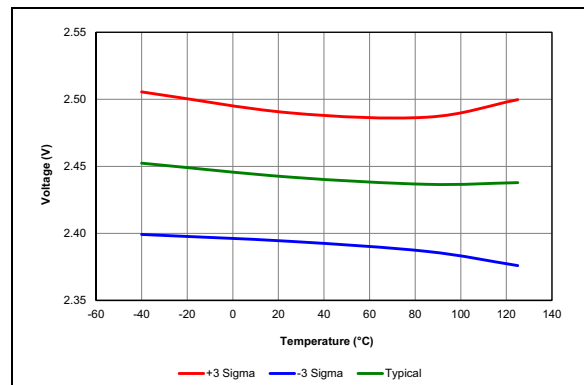


**FIGURE 46-78:** Brown-Out Reset Voltage, Trip Point (BORV = 10).

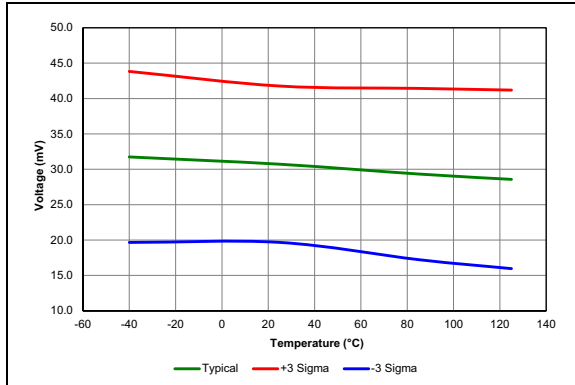


**FIGURE 46-79:** Brown-Out Reset Hysteresis, Trip Point (BORV = 10).

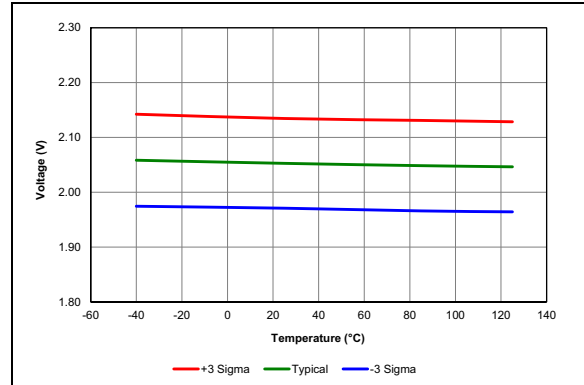


**FIGURE 46-82:** LPBOR Reset Voltage.

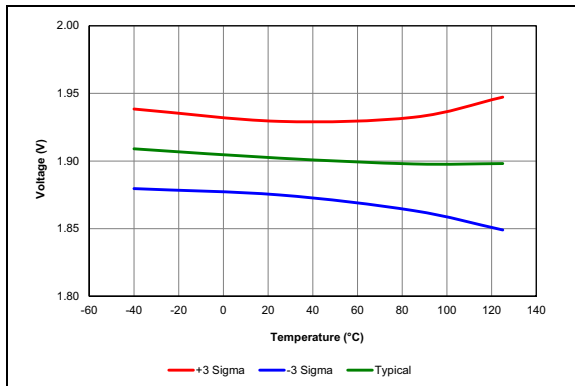


**FIGURE 46-80:** Brown-Out Reset Voltage, Trip Point (BORV = 11), PIC18LF25/26K83 Only.

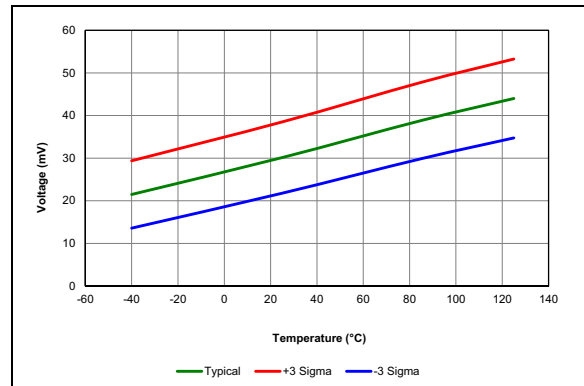


**FIGURE 46-83:** LPBOR Reset Hysteresis.

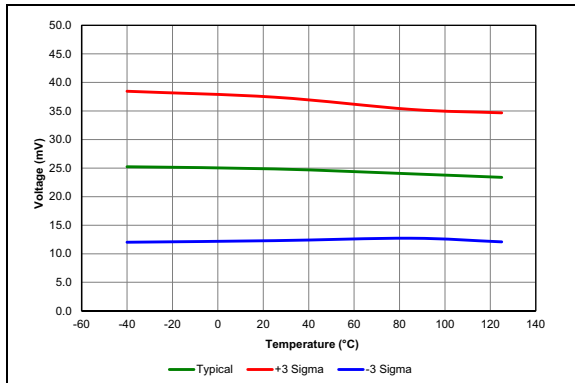


**FIGURE 46-81:** Brown-Out Reset Hysteresis, Trip Point (BORV = 11), PIC18LF25/26K83 Only.

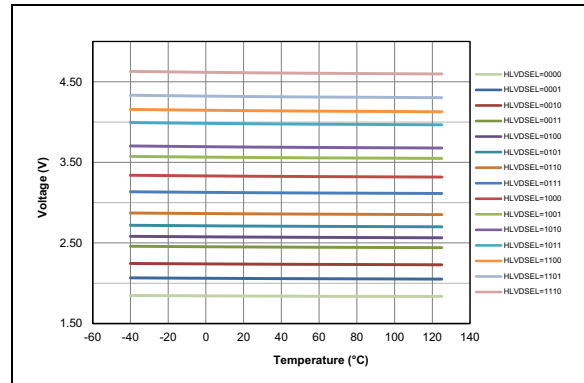


**FIGURE 46-84:** High/Low-Voltage Detect Trip Voltage.

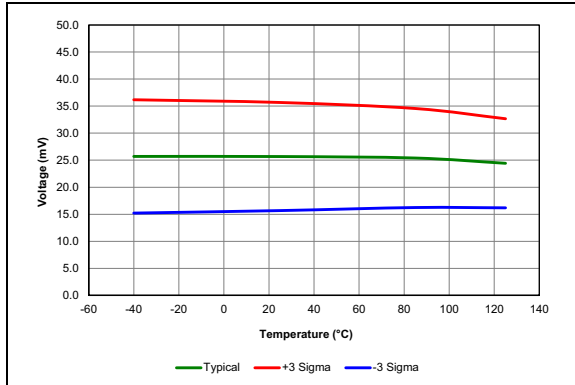


**FIGURE 46-85:** High/Low-Voltage Detect Hysteresis.

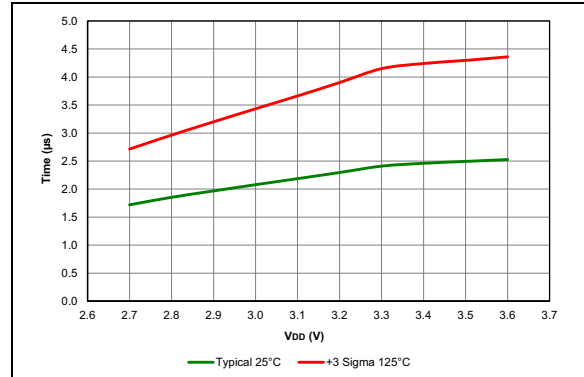


**FIGURE 46-88:** BOR Response Time, PIC18LF25/26K83 Only.

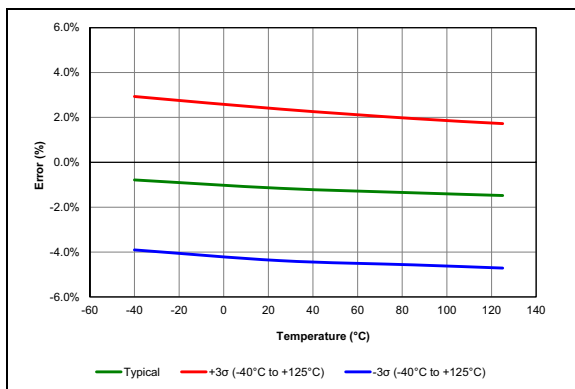


**FIGURE 46-86:** High/Low-Voltage Detect Trip Voltage, Typical Error (HLVDSEL[3:0] = 0001).

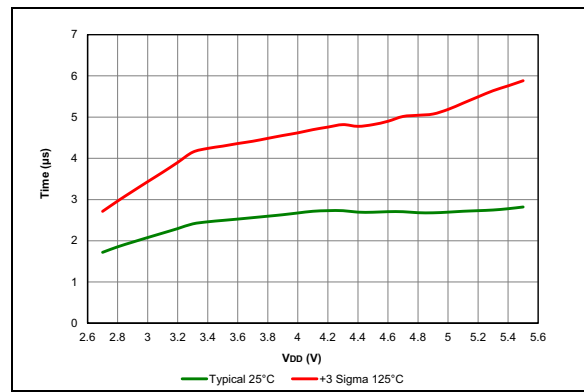


**FIGURE 46-89:** BOR Response Time, PIC18F25/26K83 Only.

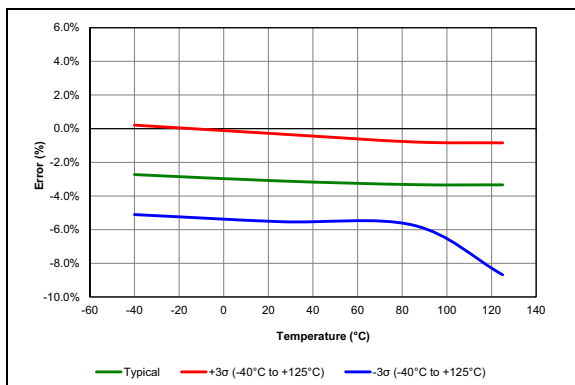


**FIGURE 46-87:** High/Low-Voltage Detect Trip Voltage, Typical Error (HLVDSEL[3:0] = 0000).

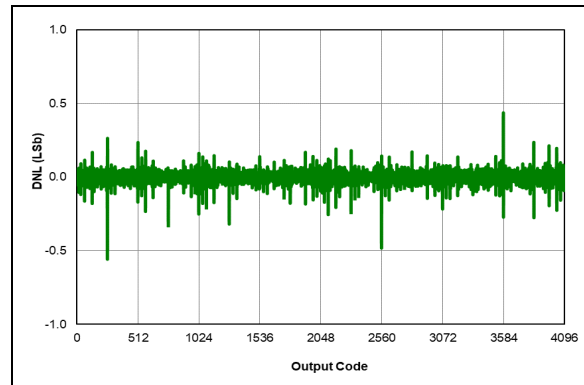


**FIGURE 46-90:** ADC 12-Bit Mode, Single-Ended, Typical DNL, VDD = 3.0V, VREF = 3.0V, TAD = 0.5 µs, 25°C, All devices.



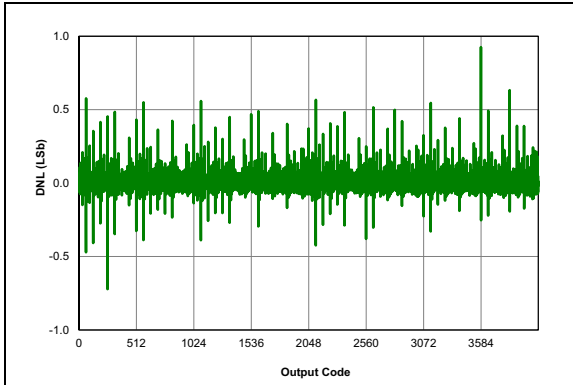


**FIGURE 46-91:** ADC 12-bit Mode, Single-Ended DNL,  $V_{DD} = 3.0V$ ,  $V_{REF} = 3.0V$ ,  $T_{AD} = 1 \mu S$ , CP OFF,  $25^{\circ}C$ .

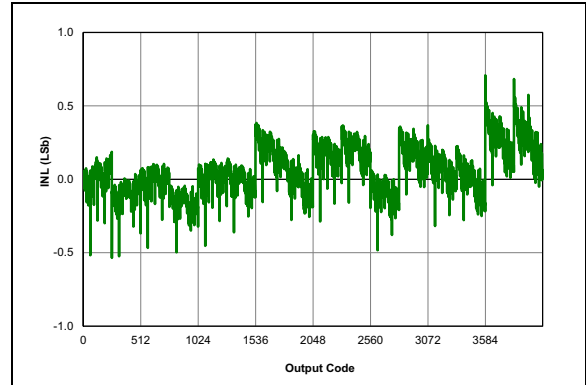


**FIGURE 46-94:** ADC 12-bit Mode, Single-Ended INL,  $V_{DD} = 3.0V$ ,  $V_{REF} = 3.0V$ ,  $T_{AD} = 1 \mu S$ , CP OFF,  $25^{\circ}C$ .

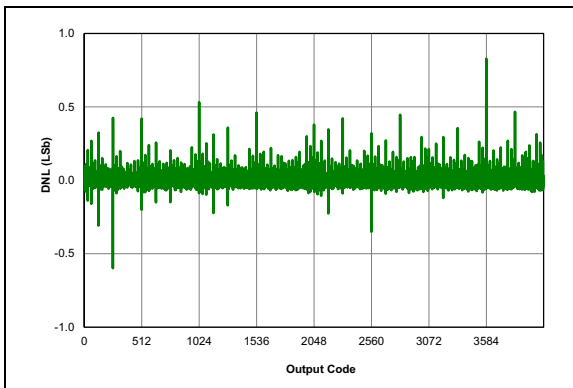


**FIGURE 46-92:** ADC 12-bit Mode, Single-Ended DNL,  $V_{DD} = 3.0V$ ,  $V_{REF} = 3.0V$ ,  $T_{AD} = 1 \mu S$ , CP ON,  $25^{\circ}C$ .

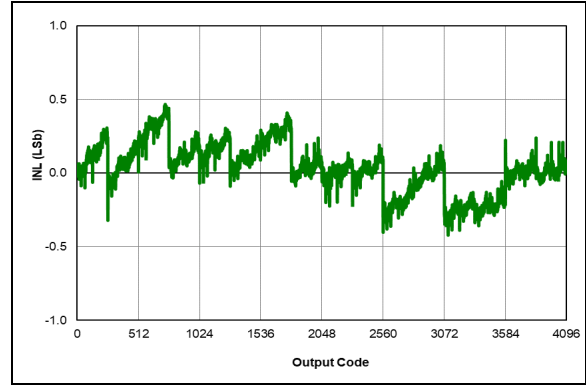


**FIGURE 46-95:** ADC 12-bit Mode, Single-Ended, Typical INL,  $V_{DD} = 3.0V$ ,  $V_{REF} = 3.0V$ ,  $T_{AD} = 0.5 \mu S$ ,  $25^{\circ}C$ , All devices.

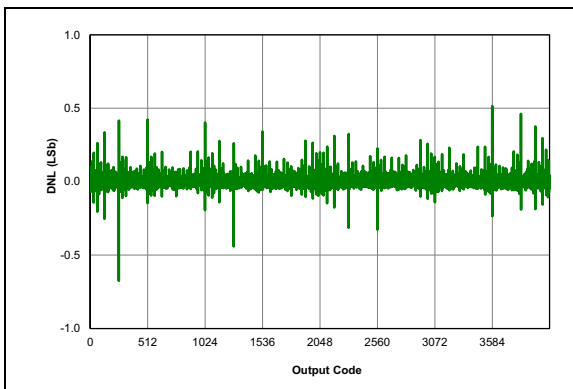


**FIGURE 46-93:** ADC 12-bit Mode, Single-Ended DNL,  $V_{DD} = 2.3V$ ,  $V_{REF} = 2.3V$ ,  $T_{AD} = 1 \mu S$ , CP ON,  $25^{\circ}C$ .

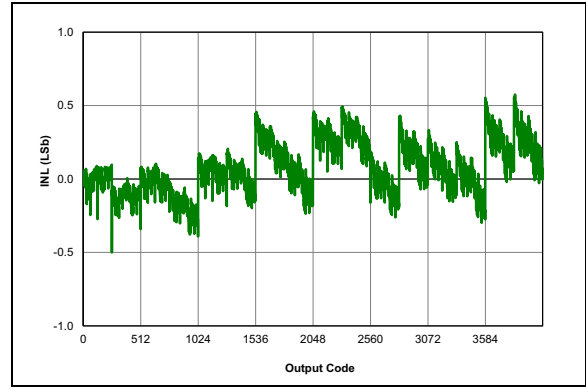


**FIGURE 46-96:** ADC 12-bit Mode, Single-Ended INL,  $V_{DD} = 3.0V$ ,  $V_{REF} = 3.0V$ ,  $T_{AD} = 1 \mu S$ , CP ON,  $25^{\circ}C$ .

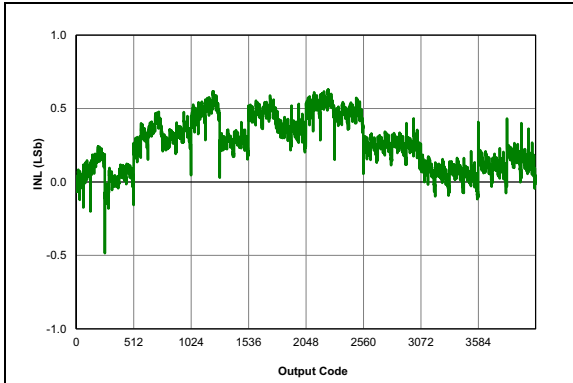


**FIGURE 46-97:** ADC 12-bit Mode, Single-Ended INL,  $V_{DD} = 2.3V$ ,  $V_{REF} = 2.3V$ ,  $T_{AD} = 1 \mu S$ , CP ON,  $25^{\circ}C$ .

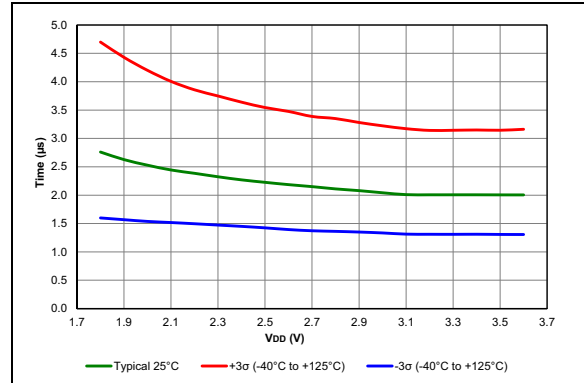


**FIGURE 46-100:** ADC RC Oscillator Period, PIC18LF25/26K83 Only.

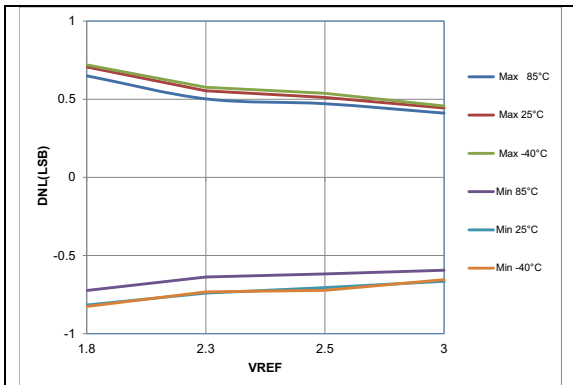


**FIGURE 46-98:** ADC 12-bit Mode, Single-Ended Typical DNL,  $V_{DD} = 3.0V$ ,  $T_{AD} = 1 \mu S$ , CP ON

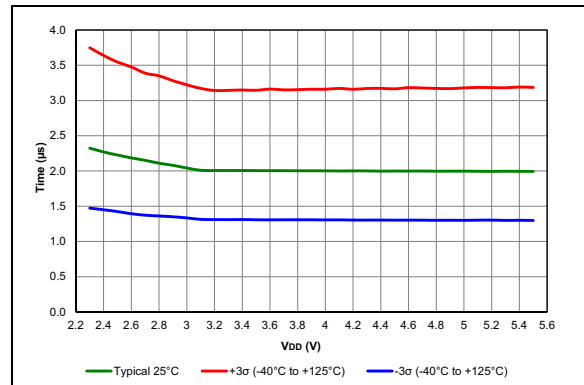


**FIGURE 46-101:** ADC RC Oscillator Period, PIC18F25/26K83 Only.

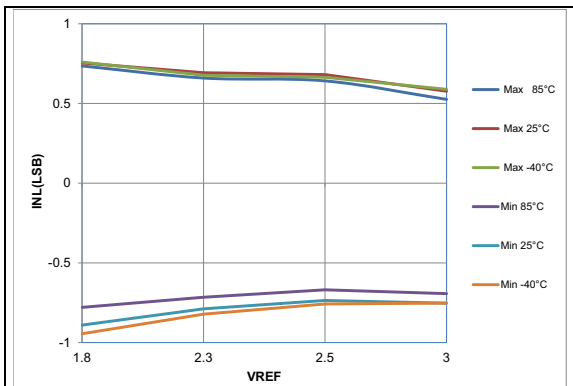


**FIGURE 46-99:** ADC 12-bit Mode, Single-Ended Typical INL,  $V_{DD} = 3.0V$ ,  $T_{AD} = 1 \mu S$ , CP ON.

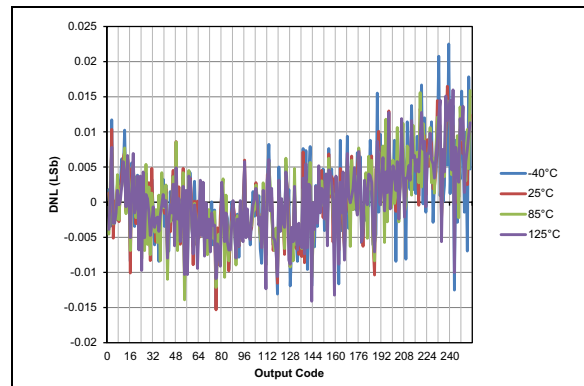


**FIGURE 46-102:** Typical DAC DNL Error,  $V_{DD} = 3.0V$ ,  $V_{REF} = \text{External } 3.0V$ .

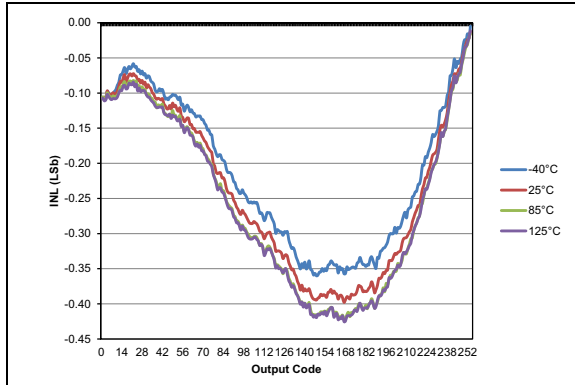


**FIGURE 46-103:** Typical DAC INL Error,  $V_{DD} = 3.0V$ ,  $V_{REF} = \text{External } 3.0V$ .

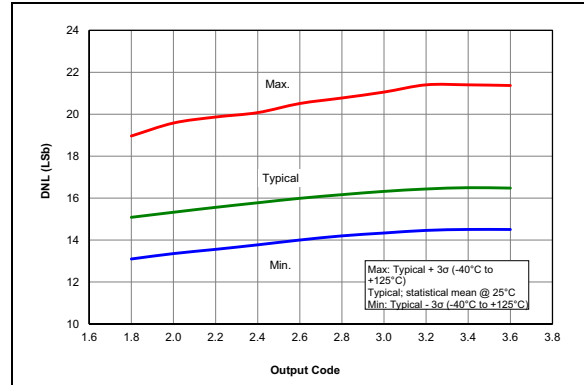


**FIGURE 46-106:** DAC INL Error,  $V_{DD} = 3.0V$ , PIC18LF25/26K83 Only.

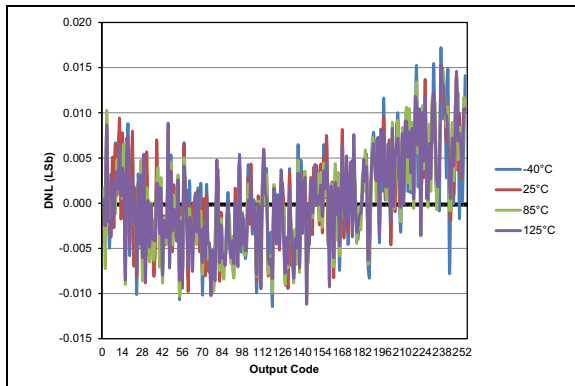


**FIGURE 46-104:** Typical DAC DNL Error,  $V_{DD} = 5.0V$ ,  $V_{REF} = \text{External } 5.0V$ , PIC18F25/26K83 Only.

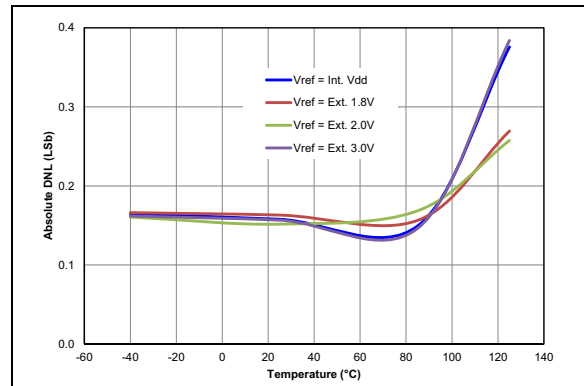


**FIGURE 46-107:** Absolute Value of DAC DNL Error,  $V_{DD} = 3.0V$ ,  $V_{REF} = V_{DD}$ .

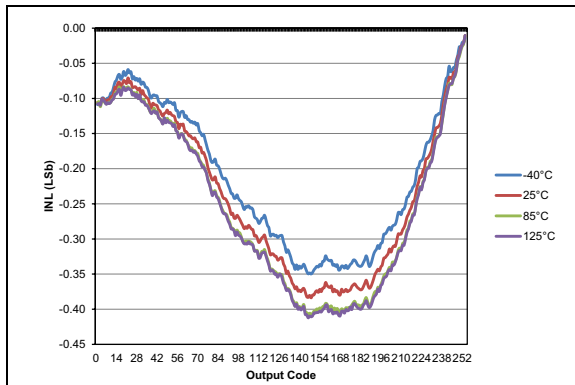


**FIGURE 46-105:** Typical DAC INL Error,  $V_{DD} = 5.0V$ ,  $V_{REF} = \text{External } 5.0V$ , PIC18F25/26K83 Only.

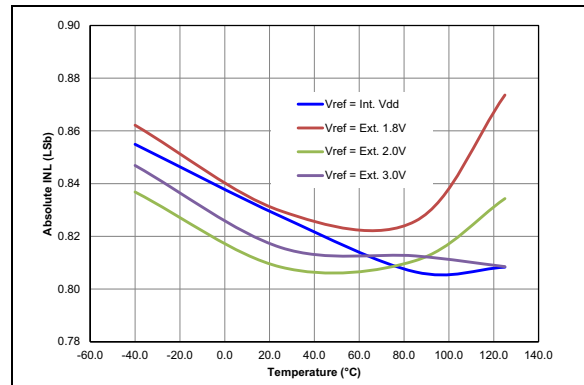


**FIGURE 46-108:** Absolute Value of DAC INL Error,  $V_{DD} = 3.0V$ ,  $V_{REF} = V_{DD}$ .

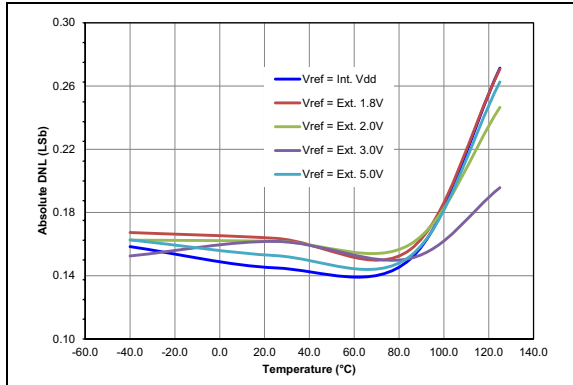


**FIGURE 46-109:** Absolute Value of DAC DNL Error,  $V_{DD} = 5.0V$ ,  $V_{REF} = V_{DD}$ , PIC18F25/26K83 Only

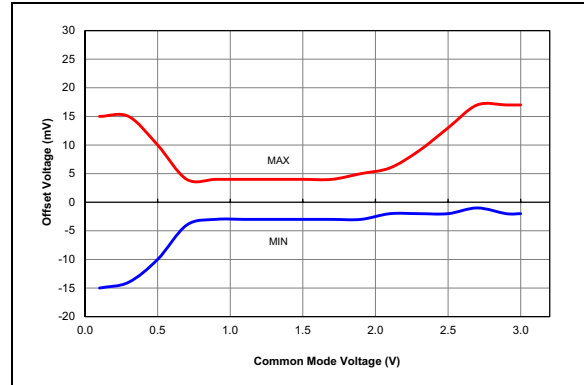


**FIGURE 46-112:** Comparator Offset, NP Mode ( $CxSP = 1$ ),  $V_{DD} = 3.0V$ , Typical Measured Values at 25°C.

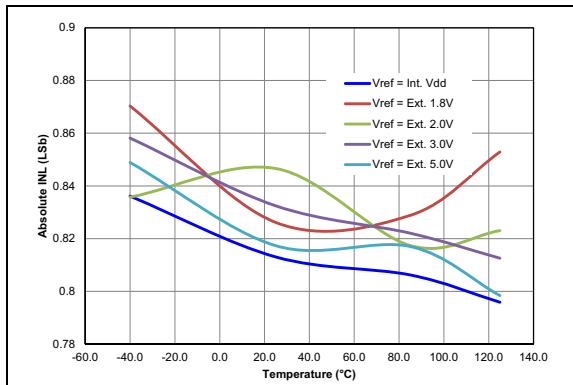


**FIGURE 46-110:** Absolute Value of DAC INL Error,  $V_{DD} = 5.0V$ ,  $V_{REF} = V_{DD}$ , PIC18F25/26K83 Only.

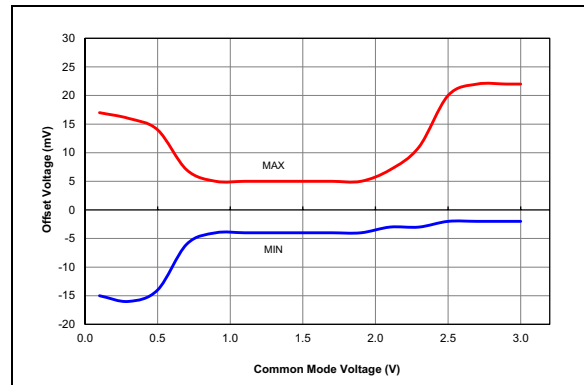


**FIGURE 46-113:** Comparator Offset, NP Mode ( $CxSP = 1$ ),  $V_{DD} = 3.0V$ , Typical Measured Values from -40°C to 125°C.

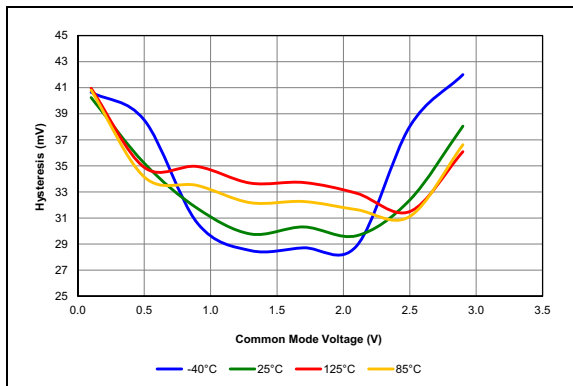


**FIGURE 46-111:** Comparator Hysteresis, NP Mode ( $CxSP = 1$ ),  $V_{DD} = 3.0V$ , Typical Measured Values.

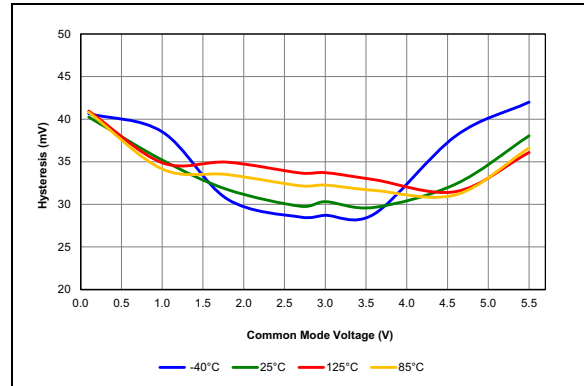


**FIGURE 46-114:** Comparator Hysteresis, NP Mode ( $CxSP = 1$ ),  $V_{DD} = 5.5V$ , Typical Measured Values, PIC18F25/26K83 Only.

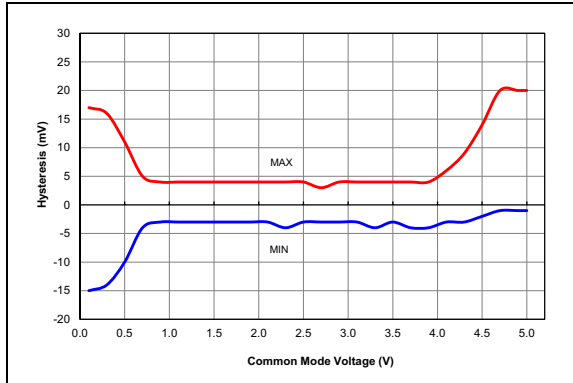


**FIGURE 46-115:** Comparator Offset, NP Mode ( $CxSP = 1$ ),  $V_{DD} = 5.0V$ , Typical Measured Values at 25°C, PIC18F25/26K83 Only

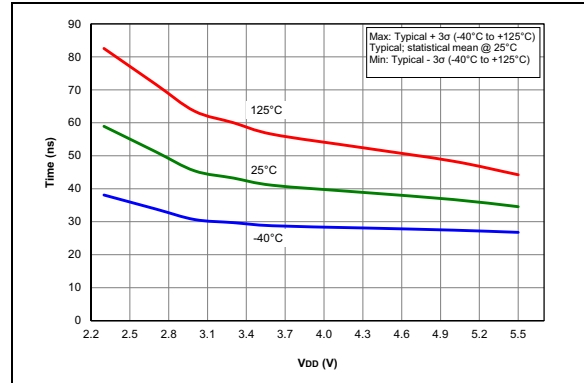


**FIGURE 46-118:** Comparator Response Time Over Voltage, NP Mode ( $CxSP = 1$ ), Typical Measured Values, PIC18F25/26K83 Only

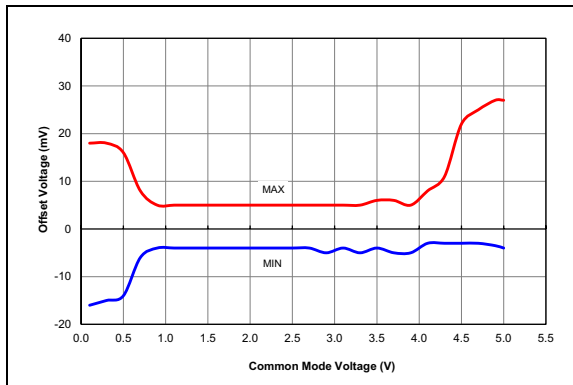


**FIGURE 46-116:** Comparator Offset, NP Mode ( $CxSP = 1$ ),  $V_{DD} = 5.5V$ , Typical Measured Values from -40°C to 125°C, PIC18F25/26K83 Only.

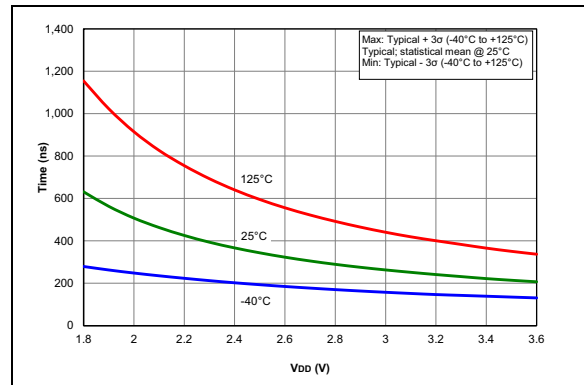


**FIGURE 46-119:** Comparator Output Filter Delay Time Over Temp., NP Mode ( $CxSP = 1$ ), Typical Measured Values, PIC18LF25/26K83 Only.

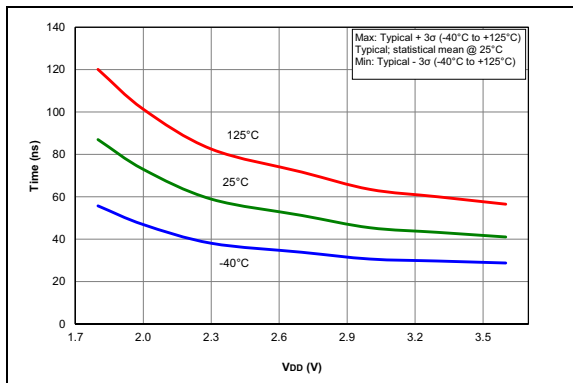


**FIGURE 46-117:** Comparator Response Time Over Voltage, NP Mode ( $CxSP = 1$ ), Typical Measured Values, PIC18LF25/26K83 Only.

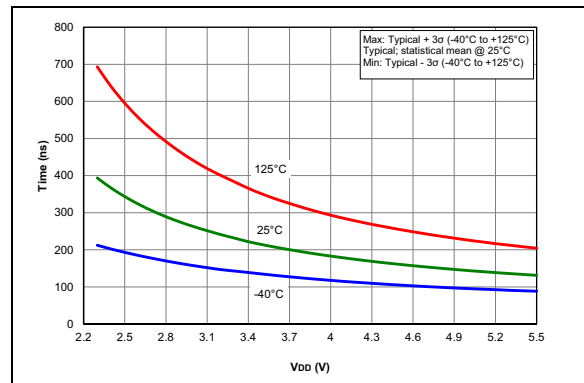


**FIGURE 46-120:** Comparator Output Filter Delay Time Over Temp., NP Mode ( $CxSP = 1$ ), Typical Measured Values, PIC18F25/26K83 Only.

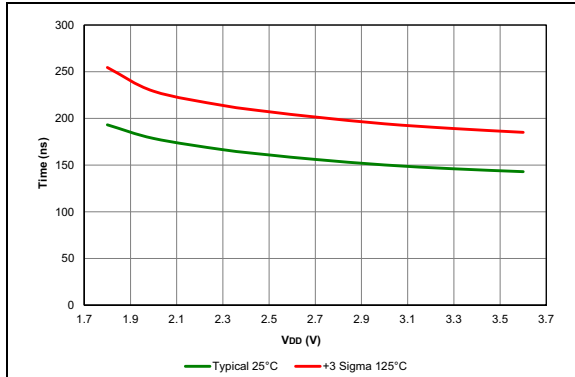


**FIGURE 46-121:** Comparator Response Time Falling Edge, PIC18LF25/26K83 Only.

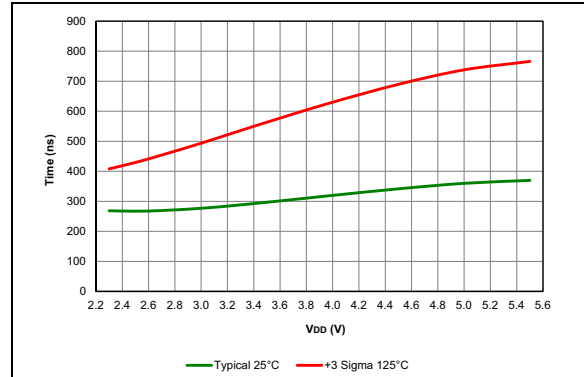


**FIGURE 46-124:** Comparator Response Time Rising Edge, PIC18F25/26K83 Only.

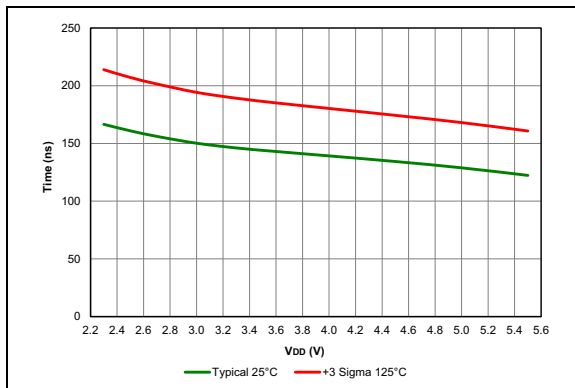


**FIGURE 46-122:** Comparator Response Time Falling Edge, PIC18F25/26K83 Only.

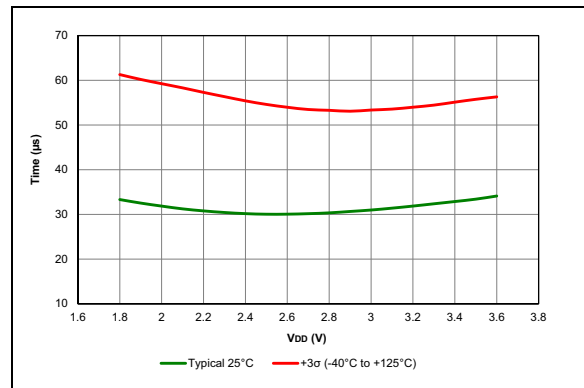


**FIGURE 46-125:** Band Gap Ready Time, PIC18LF25/26K83 Only.

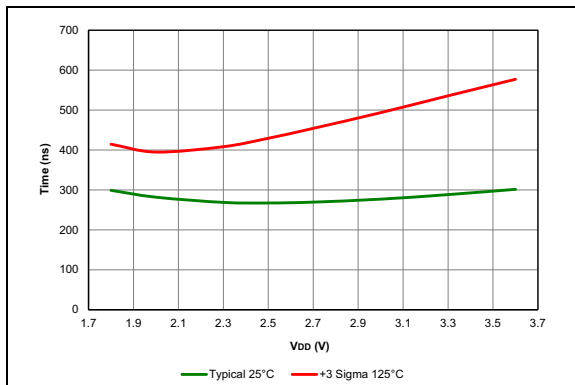


**FIGURE 46-123:** Comparator Response Time Rising Edge, PIC18LF25/26K83 Only.

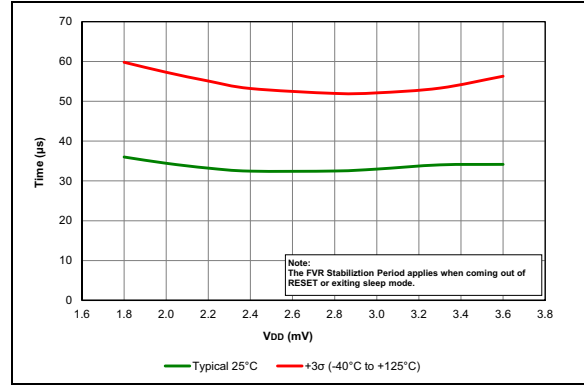


**FIGURE 46-126:** FVR Stabilization Period, PIC18LF25/26K83 Only.

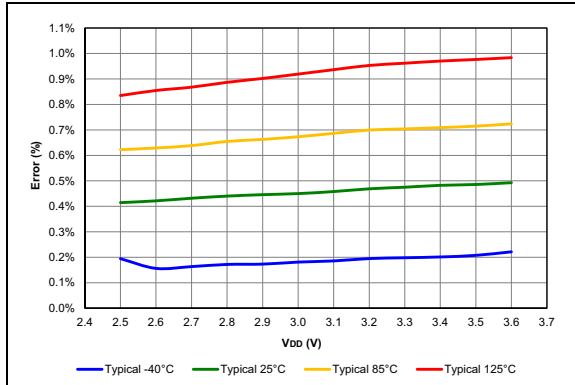


**FIGURE 46-127:** Typical FVR Voltage 1x, PIC18LF25/26K83 Only.

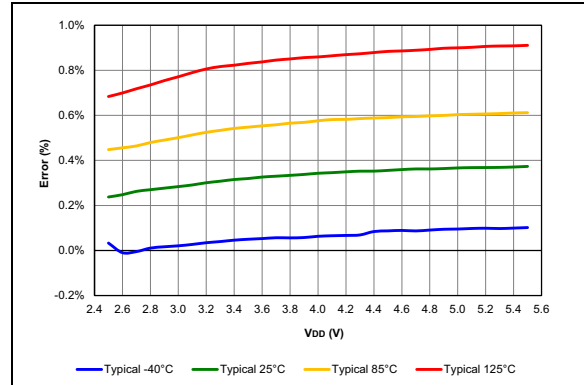


**FIGURE 46-130:** FVR Voltage Error 2x, PIC18F25/26K83 Only.

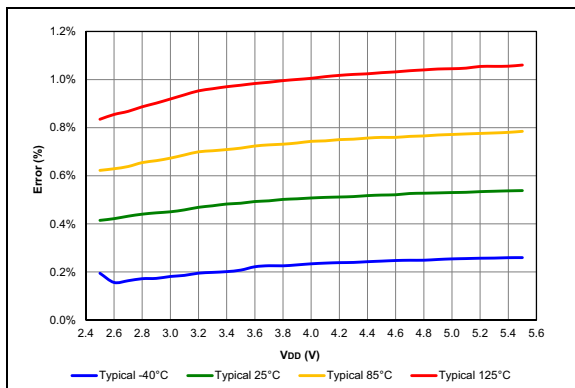


**FIGURE 46-128:** FVR Voltage Error 1x, PIC18F25/26K83 Only.

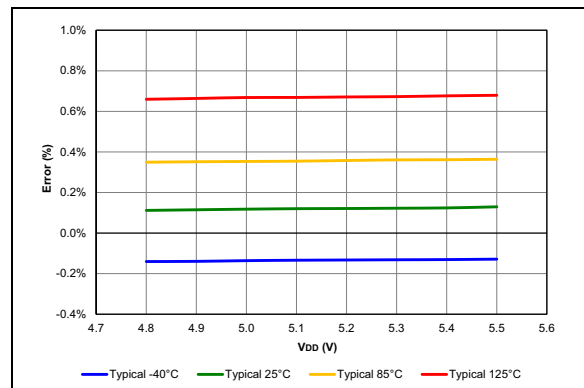


**FIGURE 46-131:** FVR Voltage Error 4x, PIC18F25/26K83 Only.

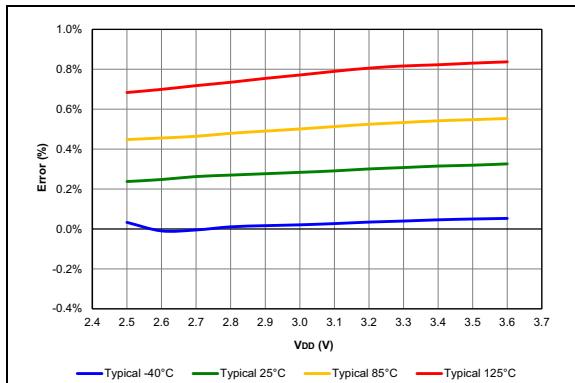


**FIGURE 46-129:** FVR Voltage Error 2x, PIC18LF25/26K83 Only.

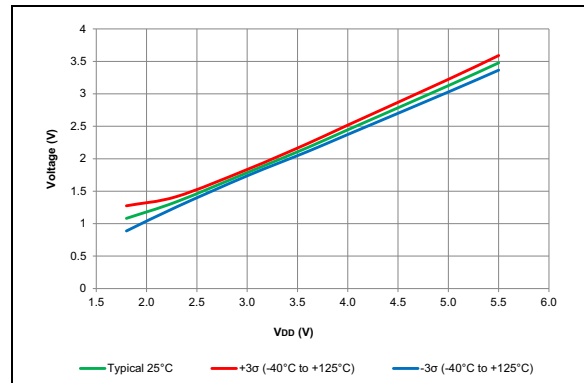


**FIGURE 46-132:** Schmitt Trigger High Values.

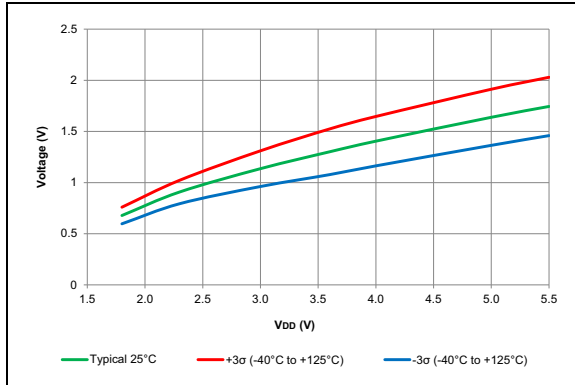


**FIGURE 46-133:** Schmitt Trigger Low Values.

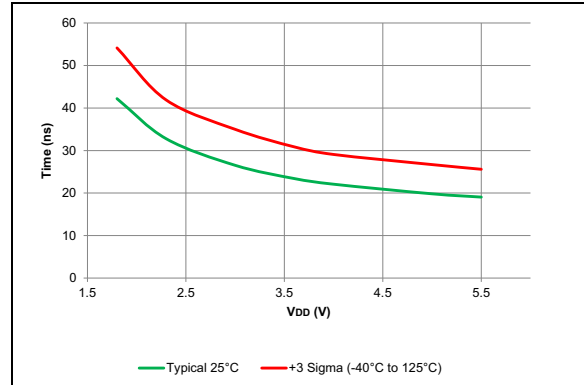


**FIGURE 46-136:** Fall Time, Slew Rate Control Enabled.

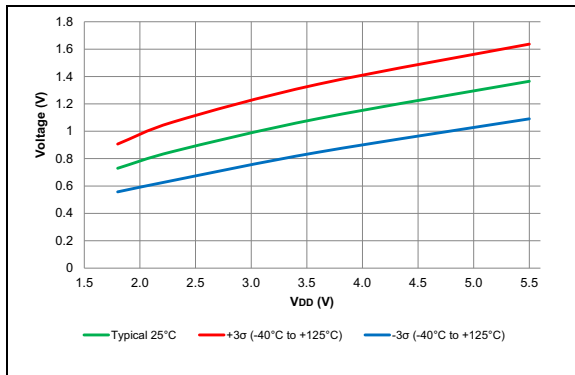


**FIGURE 46-134:** Input Level TTL.

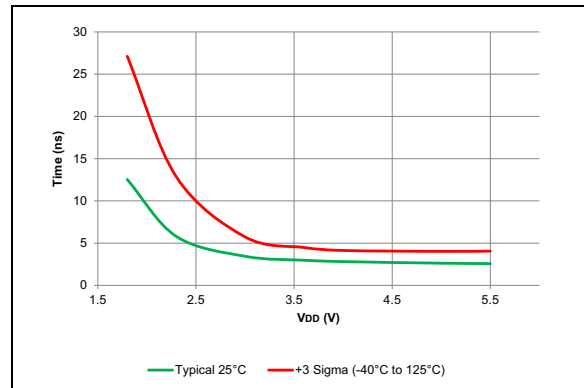


**FIGURE 46-137:** Rise Time, Slew Rate Control Disabled.

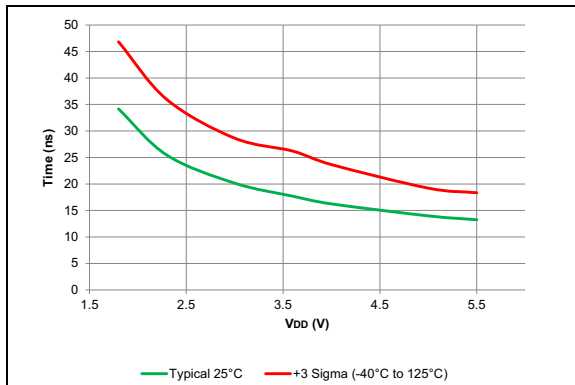


**FIGURE 46-135:** Rise Time, Slew Rate Control Enabled.

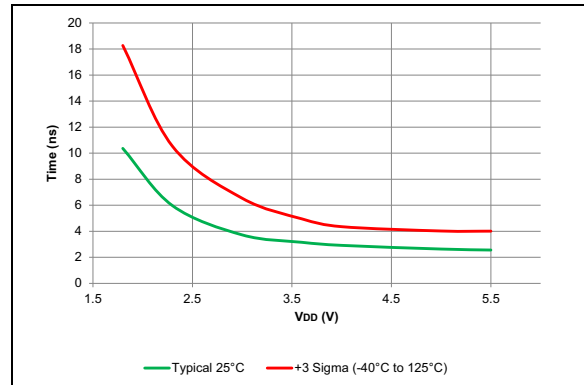


**FIGURE 46-138:** Fall Time, Slew Rate Control Disabled.



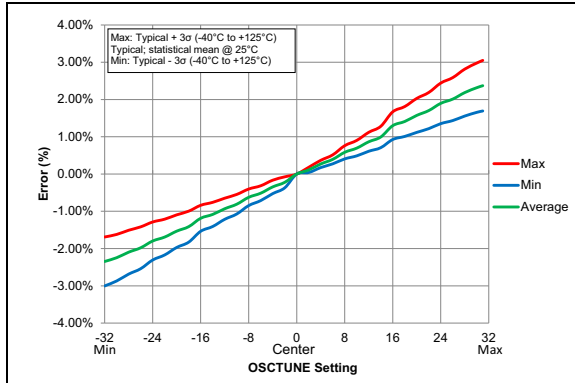


**FIGURE 46-139:** OSCTUNE Center Frequency, PIC18LF25/26K83 Only.

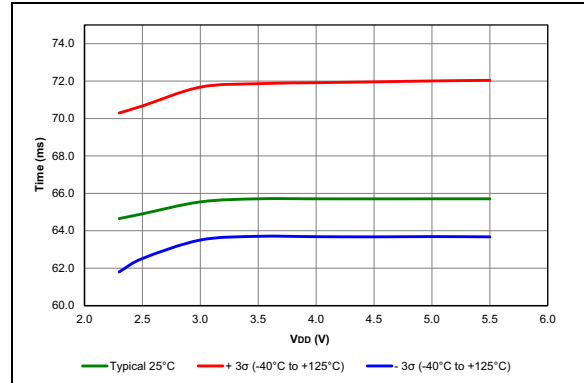


**FIGURE 46-142:** PWRT Period, PIC18F25/26K83 Only.

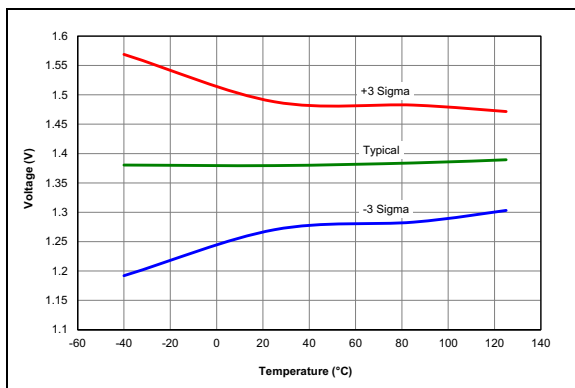


**FIGURE 46-140:** POR Release Voltage.

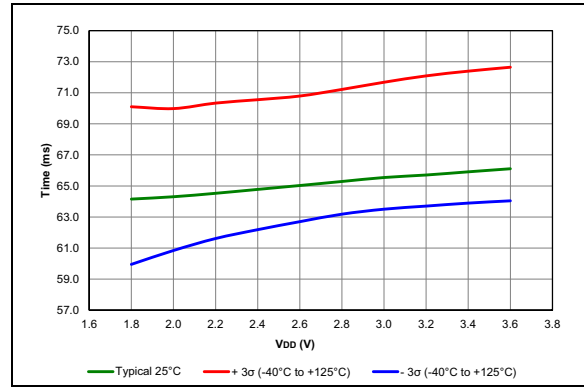


**FIGURE 46-143:** PWRT Period, PIC18LF25/26K83 Only.

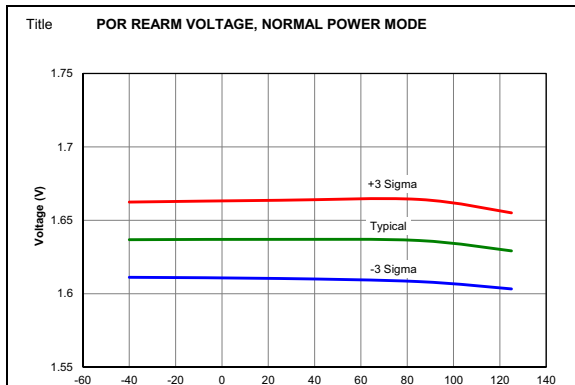


**FIGURE 46-141:** POR Rearm Voltage, NP Mode, PIC18F25/26K83 Only.

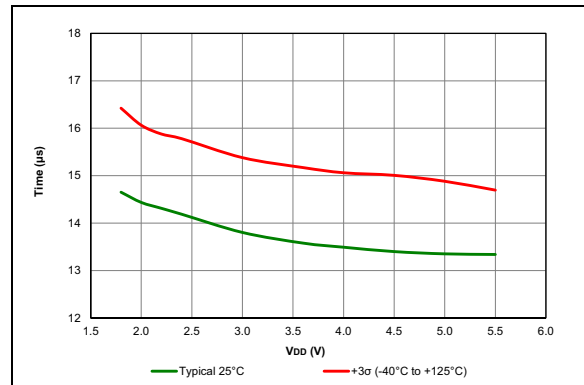


**FIGURE 46-144:** Wake from Sleep, VREGPM = 0, HFINTOSC = 4 MHz, PIC18F25/26K83 Only.

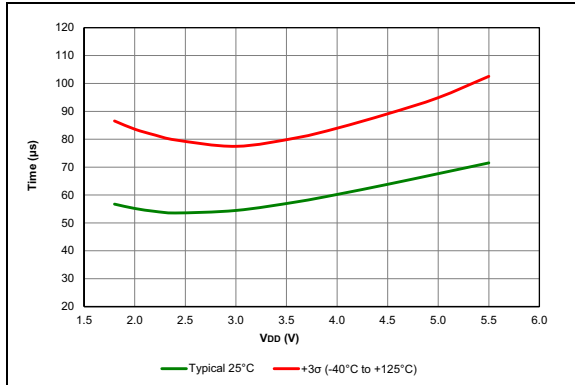


**FIGURE 46-145:** Wake from Sleep, VREGPM = 1, HFINTOSC = 4 MHz, PIC18F25/26K83 Only.

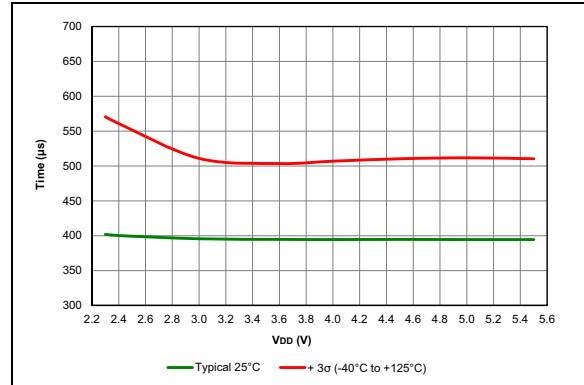


**FIGURE 46-148:** Wake from Sleep, VREGPM = 1, PIC18F25/26K83 Only.

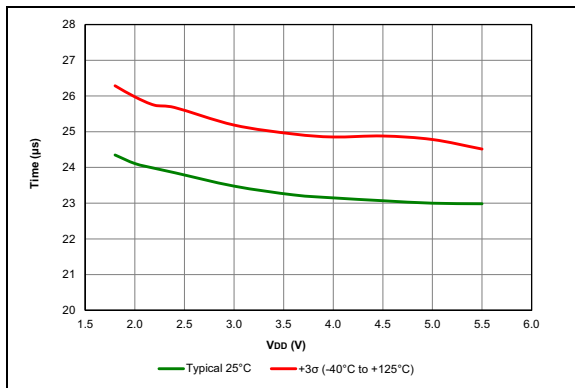


**FIGURE 46-146:** Wake from Sleep, VREGPM = 0, HFINTOSC = 16 MHz, PIC18F25/26K83 Only.

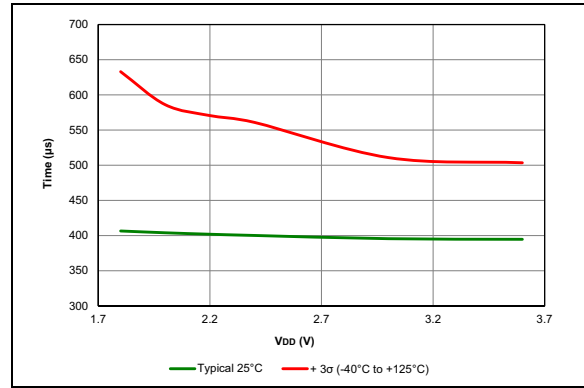


**FIGURE 46-149:** Wake from Sleep, PIC18LF25/26K83 Only.

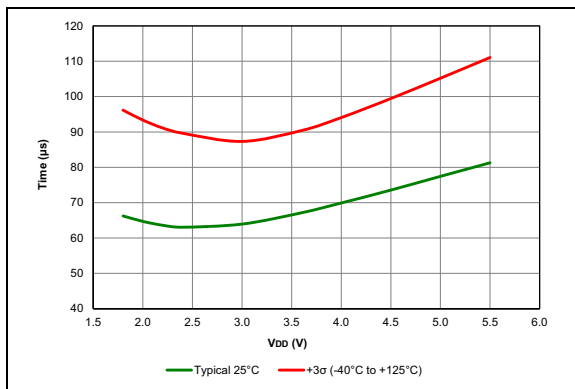


**FIGURE 46-147:** Wake from Sleep, VREGPM = 1, HFINTOSC = 16 MHz, PIC18F25/26K83 Only.

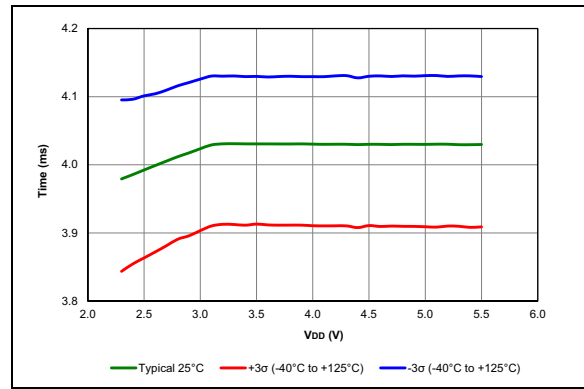


**FIGURE 46-150:** WDT Time-Out Period, PIC18F25/26K83 Only.

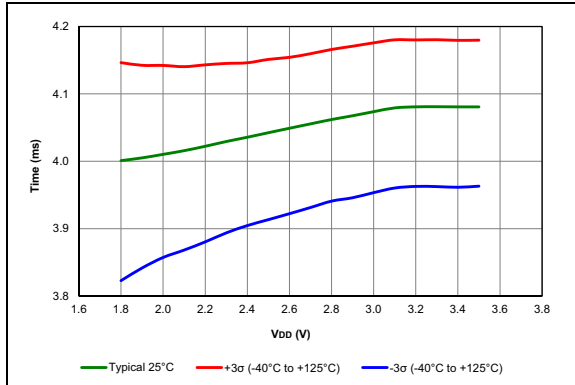


**FIGURE 46-151:** WDT Time-Out Period, PIC18LF25/26K83 Only.

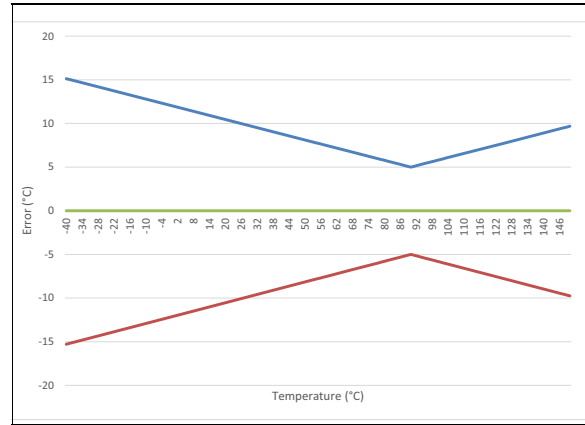


**FIGURE 46-154:** Temperature Indicator Performance Over Temperature

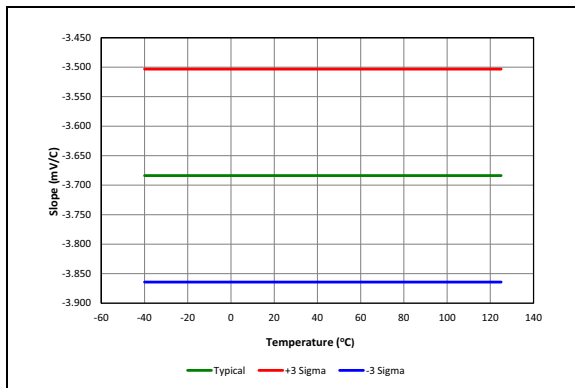


**FIGURE 46-152:** High Range Temperature Indicator Voltage Sensitivity Across Temperature.

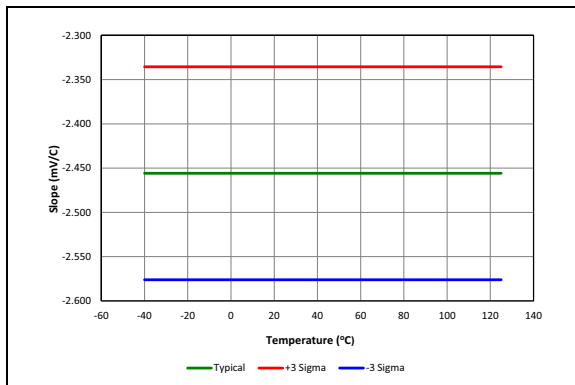
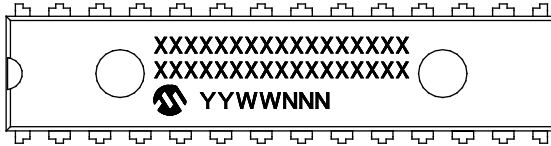


**FIGURE 46-153:** Low Range Temperature Indicator Voltage Sensitivity Across Temperature

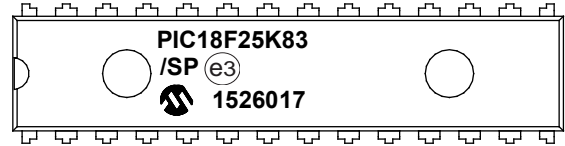
## 47.0 PACKAGING INFORMATION

### Package Marking Information

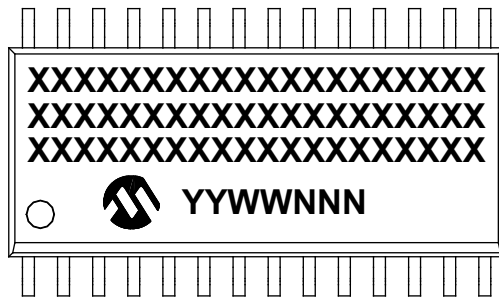
28-Lead SPDIP (.300")



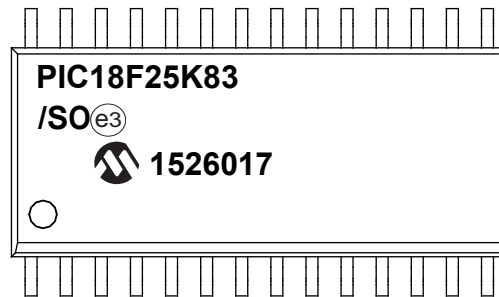
Example



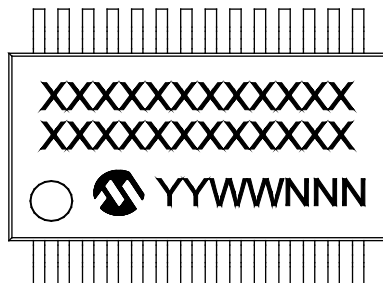
28-Lead SOIC (7.50 mm)



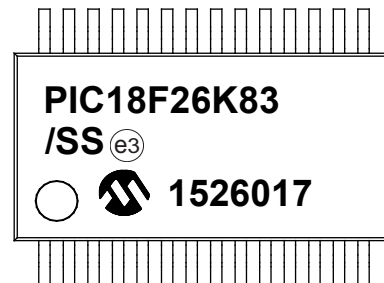
Example



28-Lead SSOP (5.30 mm)



Example



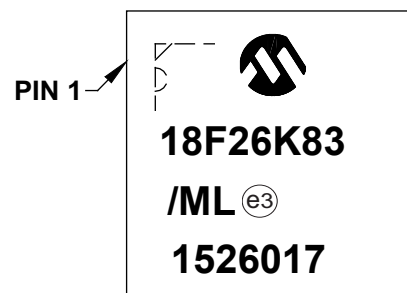
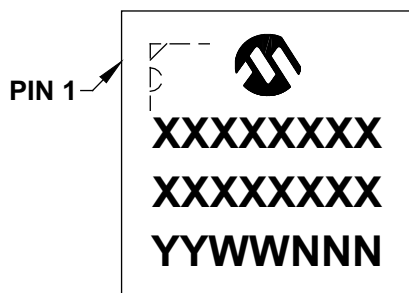
|                |        |                                                                                                                  |
|----------------|--------|------------------------------------------------------------------------------------------------------------------|
| <b>Legend:</b> | XX...X | Customer-specific information or Microchip part number                                                           |
|                | Y      | Year code (last digit of calendar year)                                                                          |
|                | YY     | Year code (last 2 digits of calendar year)                                                                       |
|                | WW     | Week code (week of January 1 is week '01')                                                                       |
|                | NNN    | Alphanumeric traceability code                                                                                   |
|                | (e3)   | Pb-free JEDEC® designator for Matte Tin (Sn)                                                                     |
|                | *      | This package is Pb-free. The Pb-free JEDEC designator (e3) can be found on the outer packaging for this package. |

**Note:** In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.

## Package Marking Information (Continued)

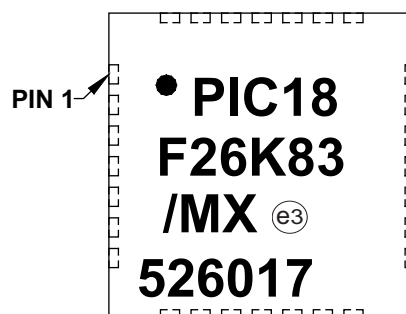
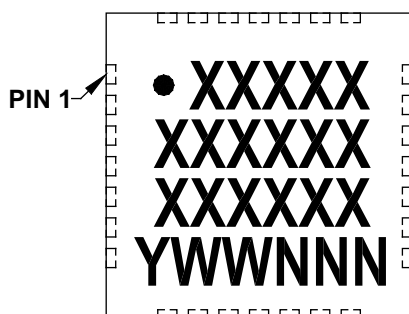
28-Lead QFN (6x6 mm)

Example



28-Lead UQFN (6x6x0.5 mm)

Example



|                |        |                                                                                                                    |
|----------------|--------|--------------------------------------------------------------------------------------------------------------------|
| <b>Legend:</b> | XX...X | Customer-specific information or Microchip part number                                                             |
|                | Y      | Year code (last digit of calendar year)                                                                            |
|                | YY     | Year code (last 2 digits of calendar year)                                                                         |
|                | WW     | Week code (week of January 1 is week '01')                                                                         |
|                | NNN    | Alphanumeric traceability code                                                                                     |
|                | (e3)   | Pb-free JEDEC <sup>®</sup> designator for Matte Tin (Sn)                                                           |
|                | *      | This package is Pb-free. The Pb-free JEDEC designator ((e3)) can be found on the outer packaging for this package. |

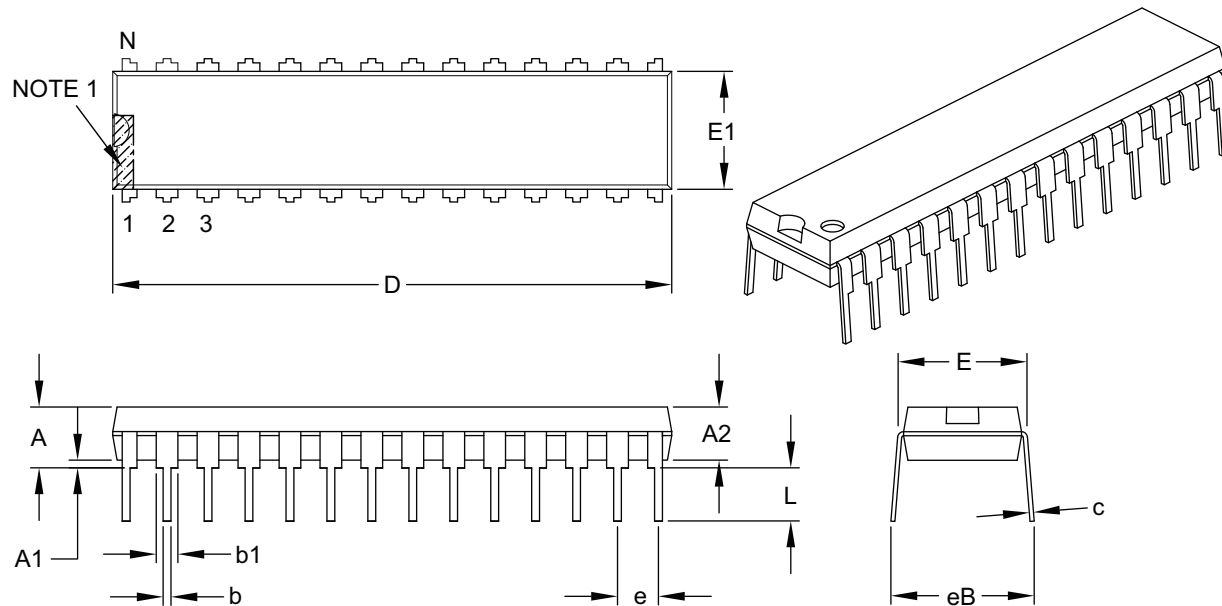
**Note:** In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.

## 47.1 Package Details

The following sections give the technical details of the packages.

### 28-Lead Skinny Plastic Dual In-Line (SP) – 300 mil Body [SPDIP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



| Units                      |    | INCHES   |       |       |
|----------------------------|----|----------|-------|-------|
| Dimension Limits           |    | MIN      | NOM   | MAX   |
| Number of Pins             | N  | 28       |       |       |
| Pitch                      | e  | .100 BSC |       |       |
| Top to Seating Plane       | A  | –        | –     | .200  |
| Molded Package Thickness   | A2 | .120     | .135  | .150  |
| Base to Seating Plane      | A1 | .015     | –     | –     |
| Shoulder to Shoulder Width | E  | .290     | .310  | .335  |
| Molded Package Width       | E1 | .240     | .285  | .295  |
| Overall Length             | D  | 1.345    | 1.365 | 1.400 |
| Tip to Seating Plane       | L  | .110     | .130  | .150  |
| Lead Thickness             | c  | .008     | .010  | .015  |
| Upper Lead Width           | b1 | .040     | .050  | .070  |
| Lower Lead Width           | b  | .014     | .018  | .022  |
| Overall Row Spacing §      | eB | –        | –     | .430  |

**Notes:**

- Pin 1 visual index feature may vary, but must be located within the hatched area.
- § Significant Characteristic.
- Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" per side.
- Dimensioning and tolerancing per ASME Y14.5M.

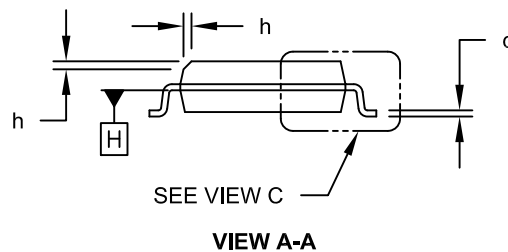
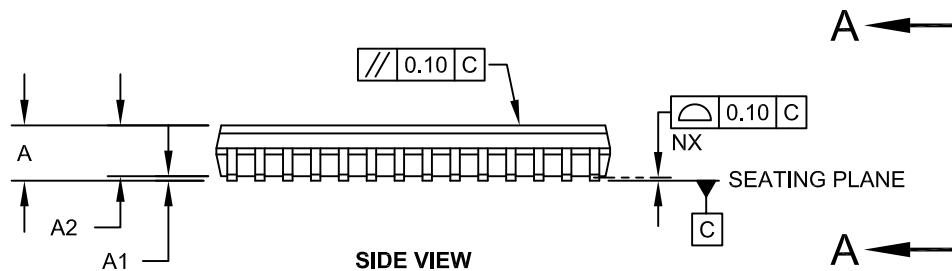
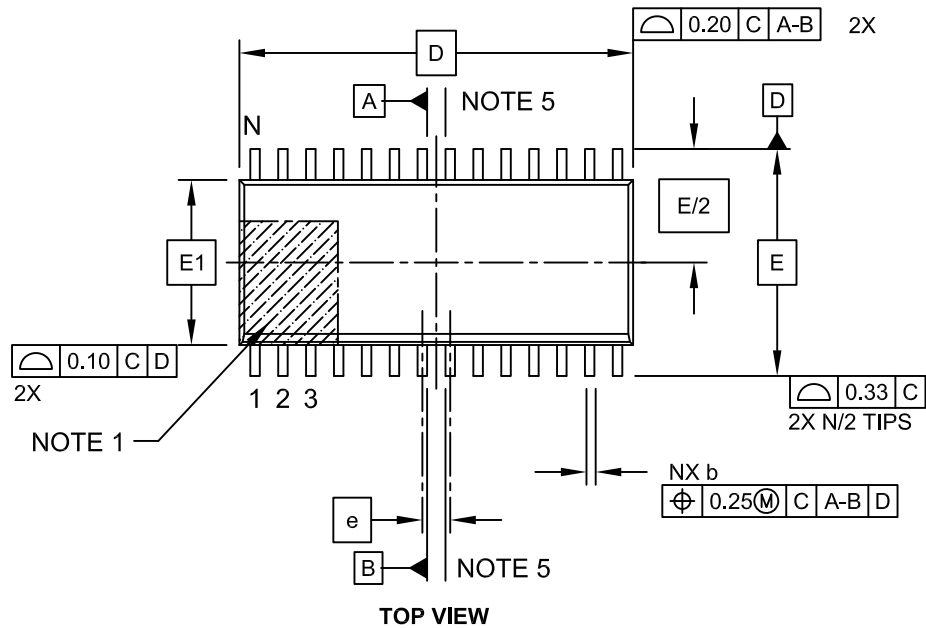
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing C04-070B

# PIC18(L)F25/26K83

## 28-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

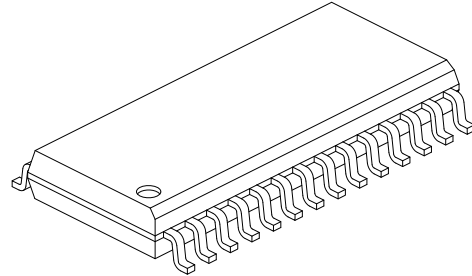
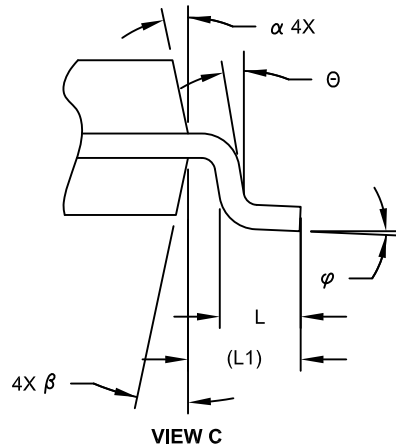
**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



Microchip Technology Drawing C04-052C Sheet 1 of 2

## 28-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



| Dimension                | Units | MILLIMETERS |     |      |
|--------------------------|-------|-------------|-----|------|
|                          |       | MIN         | NOM | MAX  |
| Number of Pins           | N     | 28          |     |      |
| Pitch                    | e     | 1.27 BSC    |     |      |
| Overall Height           | A     | -           | -   | 2.65 |
| Molded Package Thickness | A2    | 2.05        | -   | -    |
| Standoff §               | A1    | 0.10        | -   | 0.30 |
| Overall Width            | E     | 10.30 BSC   |     |      |
| Molded Package Width     | E1    | 7.50 BSC    |     |      |
| Overall Length           | D     | 17.90 BSC   |     |      |
| Chamfer (Optional)       | h     | 0.25        | -   | 0.75 |
| Foot Length              | L     | 0.40        | -   | 1.27 |
| Footprint                | L1    | 1.40 REF    |     |      |
| Lead Angle               | Θ     | 0°          | -   | -    |
| Foot Angle               | φ     | 0°          | -   | 8°   |
| Lead Thickness           | c     | 0.18        | -   | 0.33 |
| Lead Width               | b     | 0.31        | -   | 0.51 |
| Mold Draft Angle Top     | α     | 5°          | -   | 15°  |
| Mold Draft Angle Bottom  | β     | 5°          | -   | 15°  |

### Notes:

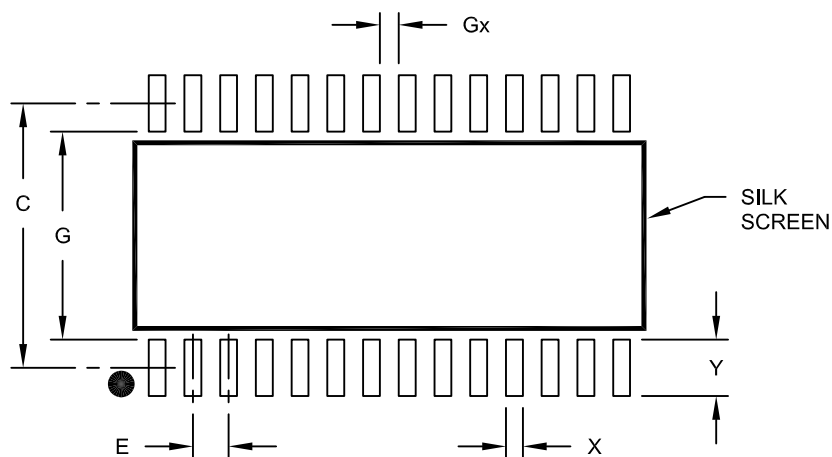
- Pin 1 visual index feature may vary, but must be located within the hatched area.
- § Significant Characteristic
- Dimension D does not include mold flash, protrusions or gate burrs, which shall not exceed 0.15 mm per end. Dimension E1 does not include interlead flash or protrusion, which shall not exceed 0.25 mm per side.
- Dimensioning and tolerancing per ASME Y14.5M
  - BSC: Basic Dimension. Theoretically exact value shown without tolerances.
  - REF: Reference Dimension, usually without tolerance, for information purposes only.
- Datums A & B to be determined at Datum H.

Microchip Technology Drawing C04-052C Sheet 2 of 2



## 28-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



### RECOMMENDED LAND PATTERN

| Units                    |    | MILLIMETERS |      |      |
|--------------------------|----|-------------|------|------|
| Dimension Limits         |    | MIN         | NOM  | MAX  |
| Contact Pitch            | E  | 1.27 BSC    |      |      |
| Contact Pad Spacing      | C  |             | 9.40 |      |
| Contact Pad Width (X28)  | X  |             |      | 0.60 |
| Contact Pad Length (X28) | Y  |             |      | 2.00 |
| Distance Between Pads    | Gx | 0.67        |      |      |
| Distance Between Pads    | G  | 7.40        |      |      |

#### Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

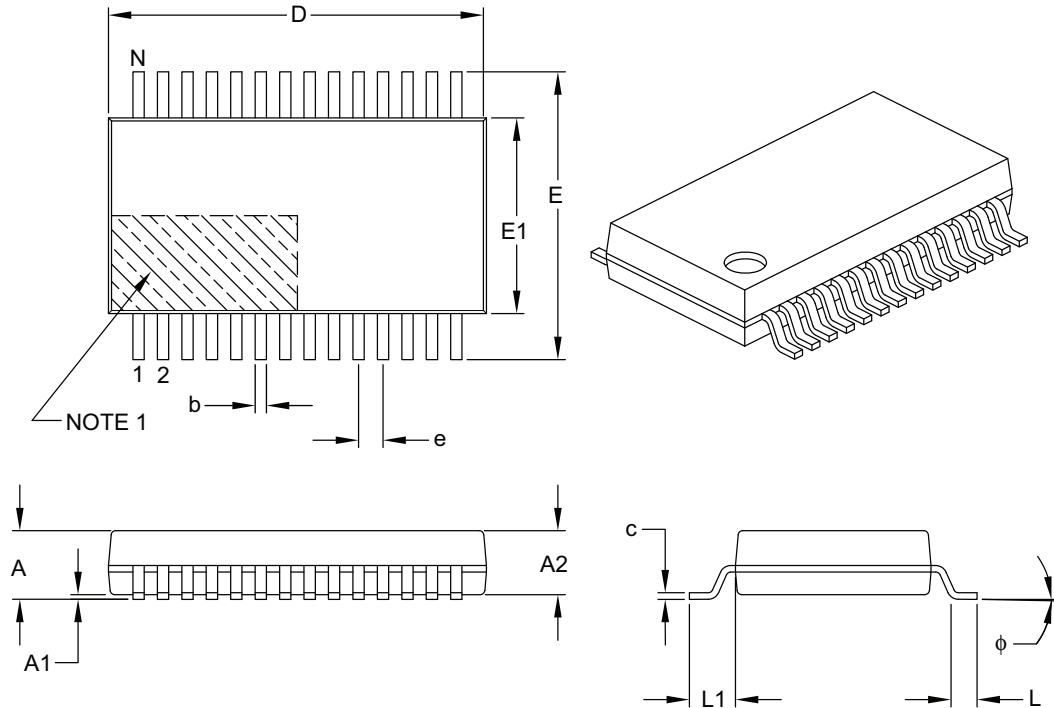
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2052A

# PIC18(L)F25/26K83

## 28-Lead Plastic Shrink Small Outline (SS) – 5.30 mm Body [SSOP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



| Units                    |    | MILLIMETERS |       |       |
|--------------------------|----|-------------|-------|-------|
| Dimension Limits         |    | MIN         | NOM   | MAX   |
| Number of Pins           | N  | 28          |       |       |
| Pitch                    | e  | 0.65 BSC    |       |       |
| Overall Height           | A  | –           | –     | 2.00  |
| Molded Package Thickness | A2 | 1.65        | 1.75  | 1.85  |
| Standoff                 | A1 | 0.05        | –     | –     |
| Overall Width            | E  | 7.40        | 7.80  | 8.20  |
| Molded Package Width     | E1 | 5.00        | 5.30  | 5.60  |
| Overall Length           | D  | 9.90        | 10.20 | 10.50 |
| Foot Length              | L  | 0.55        | 0.75  | 0.95  |
| Footprint                | L1 | 1.25 REF    |       |       |
| Lead Thickness           | c  | 0.09        | –     | 0.25  |
| Foot Angle               | φ  | 0°          | 4°    | 8°    |
| Lead Width               | b  | 0.22        | –     | 0.38  |

### Notes:

- Pin 1 visual index feature may vary, but must be located within the hatched area.
- Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.20 mm per side.
- Dimensioning and tolerancing per ASME Y14.5M.

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

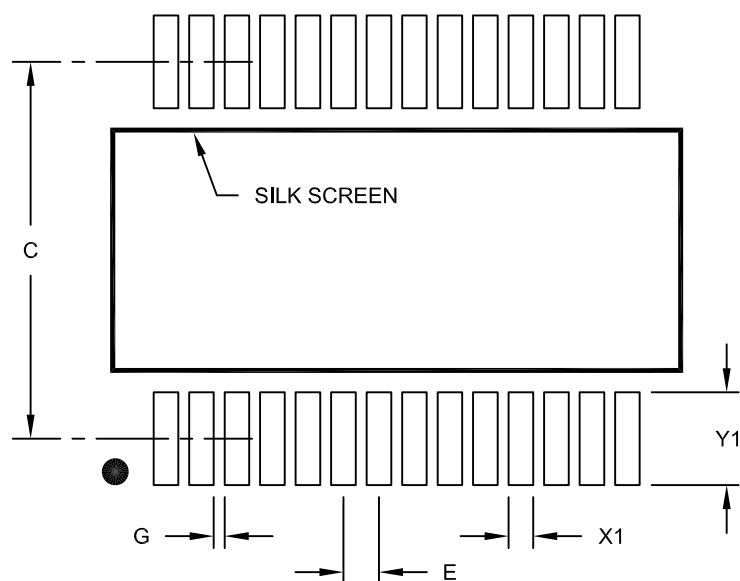
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-073B

# PIC18(L)F25/26K83

## 28-Lead Plastic Shrink Small Outline (SS) - 5.30 mm Body [SSOP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



RECOMMENDED LAND PATTERN

| Units                    |    | MILLIMETERS |      |      |
|--------------------------|----|-------------|------|------|
| Dimension Limits         |    | MIN         | NOM  | MAX  |
| Contact Pitch            | E  | 0.65 BSC    |      |      |
| Contact Pad Spacing      | C  |             | 7.20 |      |
| Contact Pad Width (X28)  | X1 |             |      | 0.45 |
| Contact Pad Length (X28) | Y1 |             |      | 1.75 |
| Distance Between Pads    | G  | 0.20        |      |      |

**Notes:**

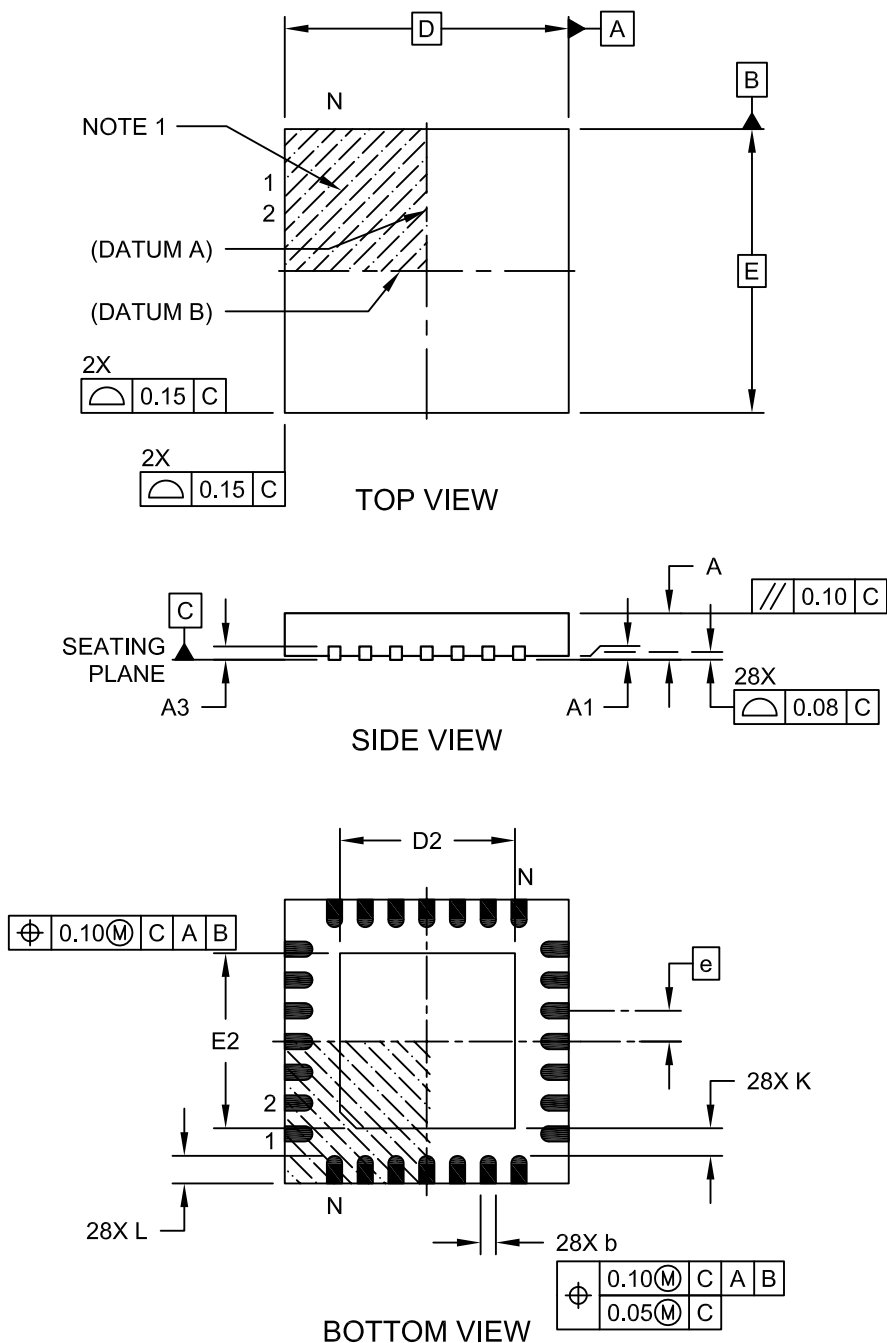
1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2073A

## 28-Lead Plastic Quad Flat, No Lead Package (ML) - 6x6 mm Body [QFN] With 0.55 mm Terminal Length

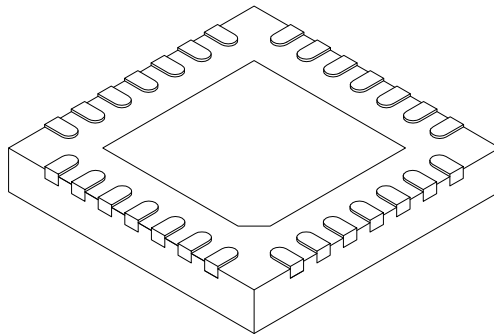
**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



Microchip Technology Drawing C04-105C Sheet 1 of 2

## 28-Lead Plastic Quad Flat, No Lead Package (ML) - 6x6 mm Body [QFN] With 0.55 mm Terminal Length

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



| Dimension               | Units  | MILLIMETERS |      |      |
|-------------------------|--------|-------------|------|------|
|                         | Limits | MIN         | NOM  | MAX  |
| Number of Pins          | N      | 28          |      |      |
| Pitch                   | e      | 0.65 BSC    |      |      |
| Overall Height          | A      | 0.80        | 0.90 | 1.00 |
| Standoff                | A1     | 0.00        | 0.02 | 0.05 |
| Terminal Thickness      | A3     | 0.20 REF    |      |      |
| Overall Width           | E      | 6.00 BSC    |      |      |
| Exposed Pad Width       | E2     | 3.65        | 3.70 | 4.20 |
| Overall Length          | D      | 6.00 BSC    |      |      |
| Exposed Pad Length      | D2     | 3.65        | 3.70 | 4.20 |
| Terminal Width          | b      | 0.23        | 0.30 | 0.35 |
| Terminal Length         | L      | 0.50        | 0.55 | 0.70 |
| Terminal-to-Exposed Pad | K      | 0.20        | -    | -    |

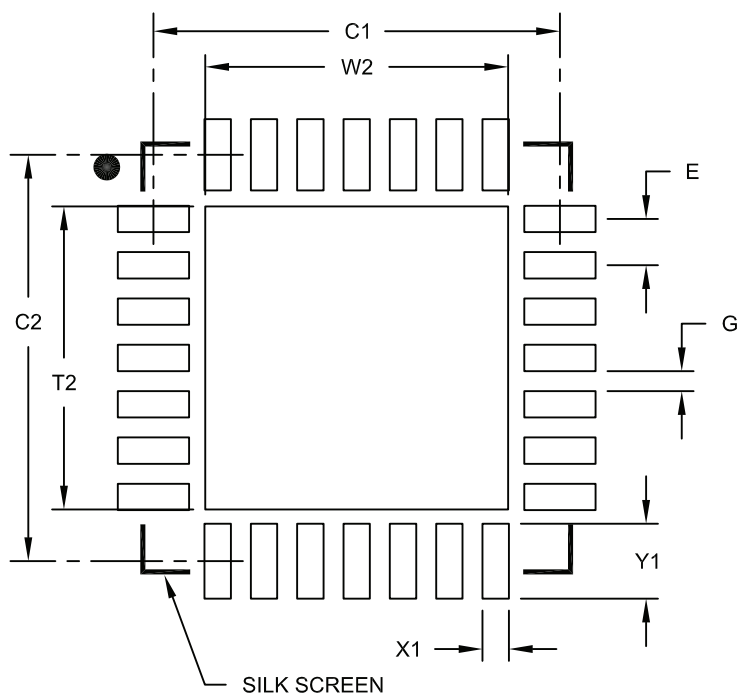
### Notes:

- Pin 1 visual index feature may vary, but must be located within the hatched area.
- Package is saw singulated
- Dimensioning and tolerancing per ASME Y14.5M.
  - BSC: Basic Dimension. Theoretically exact value shown without tolerances.
  - REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-105C Sheet 2 of 2

## 28-Lead Plastic Quad Flat, No Lead Package (ML) – 6x6 mm Body [QFN] with 0.55 mm Contact Length

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



RECOMMENDED LAND PATTERN

| Dimension Limits           | Units | MILLIMETERS |      |      |
|----------------------------|-------|-------------|------|------|
|                            |       | MIN         | NOM  | MAX  |
| Contact Pitch              | E     | 0.65 BSC    |      |      |
| Optional Center Pad Width  | W2    |             |      | 4.25 |
| Optional Center Pad Length | T2    |             |      | 4.25 |
| Contact Pad Spacing        | C1    |             | 5.70 |      |
| Contact Pad Spacing        | C2    |             | 5.70 |      |
| Contact Pad Width (X28)    | X1    |             |      | 0.37 |
| Contact Pad Length (X28)   | Y1    |             |      | 1.00 |
| Distance Between Pads      | G     | 0.20        |      |      |

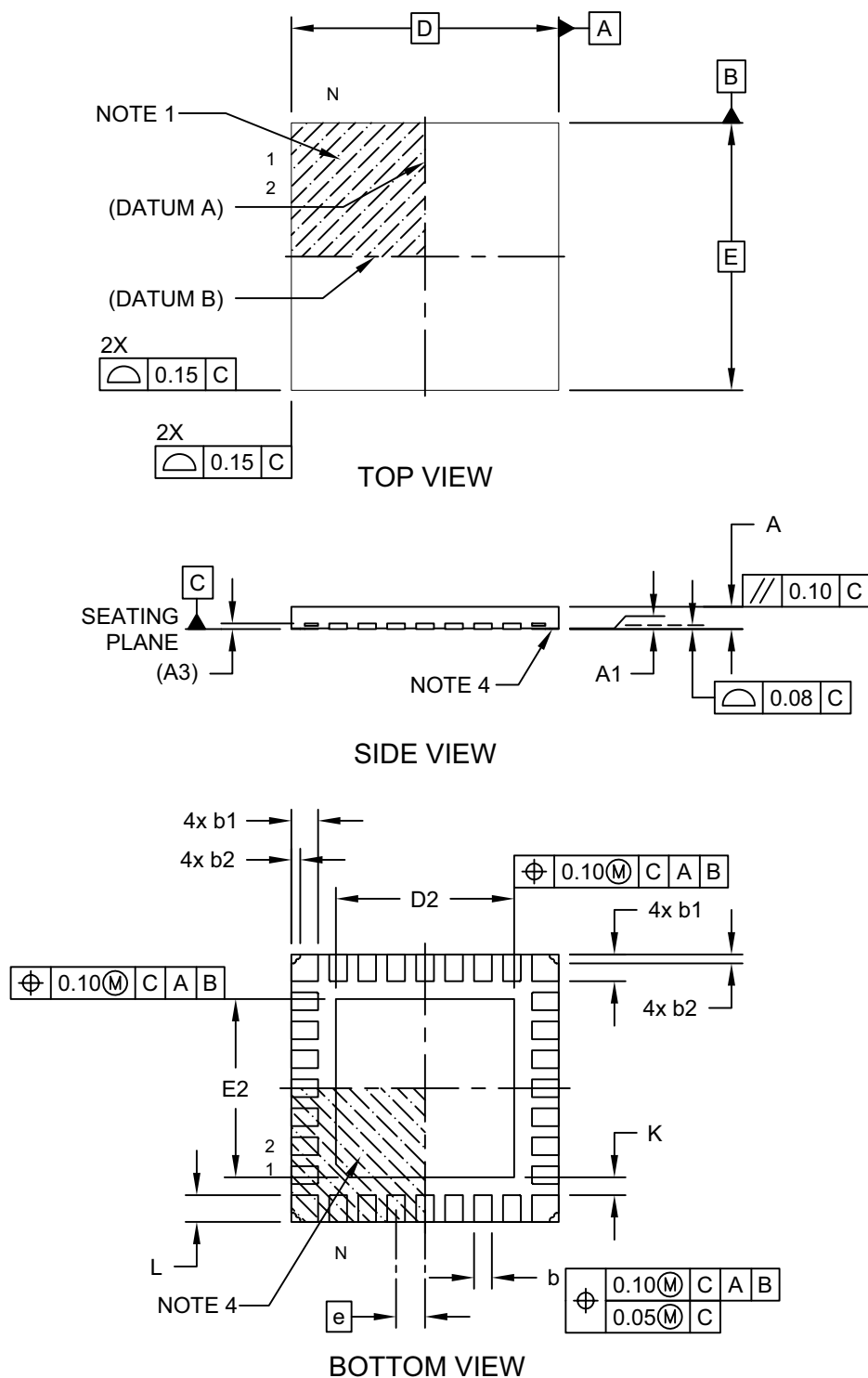
**Notes:**

1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2105A

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>

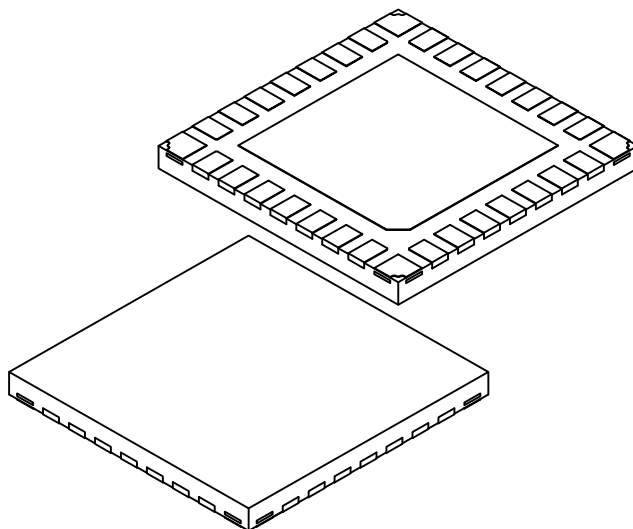


Microchip Technology Drawing C04-0209 Rev C Sheet 1 of 2

# PIC18(L)F25/26K83

## 28-Lead Plastic Quad Flat, No Lead Package (MX) - 6x6x0.5mm Body [UQFN] Ultra-Thin with 0.40 x 0.60 mm Terminal Width/Length and Corner Anchors

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



|                             |      | Units | MILLIMETERS |      |      |
|-----------------------------|------|-------|-------------|------|------|
| Dimension Limits            |      |       | MIN         | NOM  | MAX  |
| Number of Pins              | N    |       | 28          |      |      |
| Pitch                       | e    |       | 0.65 BSC    |      |      |
| Overall Height              | A    |       | 0.40        | 0.50 | 0.60 |
| Standoff                    | A1   |       | 0.00        | 0.02 | 0.05 |
| Terminal Thickness          | (A3) |       | 0.127 REF   |      |      |
| Overall Width               | E    |       | 6.00 BSC    |      |      |
| Exposed Pad Width           | E2   |       |             | 4.00 |      |
| Overall Length              | D    |       | 6.00 BSC    |      |      |
| Exposed Pad Length          | D2   |       |             | 4.00 |      |
| Terminal Width              | b    |       | 0.35        | 0.40 | 0.45 |
| Corner Pad                  | b1   |       | 0.55        | 0.60 | 0.65 |
| Corner Pad, Metal Free Zone | b2   |       | 0.15        | 0.20 | 0.25 |
| Terminal Length             | L    |       | 0.55        | 0.60 | 0.65 |
| Terminal-to-Exposed Pad     | K    |       | 0.20        | -    | -    |

### Notes:

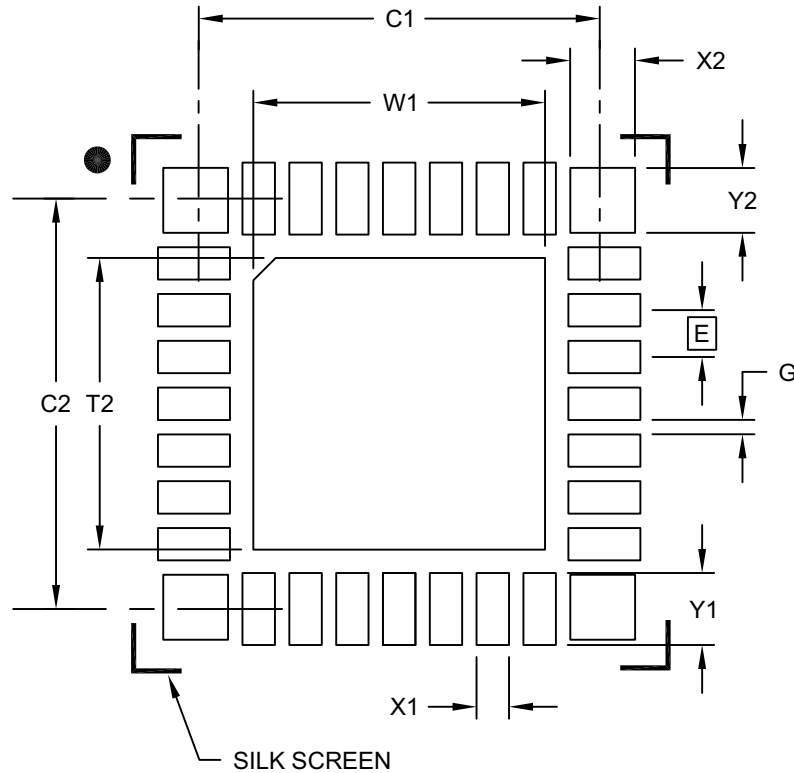
- Pin 1 visual index feature may vary, but must be located within the hatched area.
- Package is saw singulated
- Dimensioning and tolerancing per ASME Y14.5M
  - BSC: Basic Dimension. Theoretically exact value shown without tolerances.
  - REF: Reference Dimension, usually without tolerance, for information purposes only.
- Outermost portions of corner structures may vary slightly.

Microchip Technology Drawing C04-0209 Rev C Sheet 2 of 2



## 28-Lead Plastic Quad Flat, No Lead Package (MX) - 6x6 mm Body [UQFN] With 0.60mm Contact Length And Corner Anchors

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



RECOMMENDED LAND PATTERN

| Units                      |    | MILLIMETERS |      |      |
|----------------------------|----|-------------|------|------|
| Dimension Limits           |    | MIN         | NOM  | MAX  |
| Contact Pitch              | E  | 0.65 BSC    |      |      |
| Optional Center Pad Width  | W1 |             |      | 4.05 |
| Optional Center Pad Length | T2 |             |      | 4.05 |
| Contact Pad Spacing        | C1 |             | 5.70 |      |
| Contact Pad Spacing        | C2 |             | 5.70 |      |
| Contact Pad Width (X28)    | X1 |             |      | 0.45 |
| Contact Pad Length (X28)   | Y1 |             |      | 1.00 |
| Corner Pad Width (X4)      | X2 |             |      | 0.90 |
| Corner Pad Length (X4)     | Y2 |             |      | 0.90 |
| Distance Between Pads      | G  | 0.20        |      |      |

**Notes:**

1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2209B

## REVISION HISTORY

### Revision C (06/2020)

Updated Electrical Specifications data and added the temperature information to the DC and AC Characteristics Graphs and Tables chapter; other minor corrections.

### Revision B (02/2019)

Updated Digital Peripherals section in cover pages.

Updated Example 9-3; Figures 15-2, 32-1, 32-2, 36-1, and 45-1; Registers 5-1, 5-2, 5-3, 5-4, 9-8, 13-3, 13-4, 25-3, and 37-5; Sections 1.1, 4.2.3, 4.5.6, 5.7.3, 7.1, 7.2.1.1, 13.1.2.4, 13.1.6, 13.1.6.1, 13.2, 13.2.1, 15.12, 31.6, 32.2, 36.0, 36.1, 36.2, 36.2.1.2, 36.2.2, and 36.3; Tables 1, 4-3, 5-3, 6-13, 15-8, 45-1, 45-12, 45-21, 45-23 and 45-24.

Added Table 45-22: SPI Mode Requirements (Slave Mode). Added Table 45-25: Temperature Indicator Requirements.

Removed Section 36.2.1.1: Single-Point Calibration.  
Removed Section 36.6: DIA Information.

### Revision A (8/2017)

Initial release of the document.

## THE MICROCHIP WEBSITE

Microchip provides online support via our WWW site at [www.microchip.com](http://www.microchip.com). This website is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the website contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

## CUSTOMER CHANGE NOTIFICATION SERVICE

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip website at [www.microchip.com](http://www.microchip.com). Under "Support", click on "Customer Change Notification" and follow the registration instructions.

## CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or Field Application Engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

**Technical support is available through the website at:** <http://microchip.com/support>

## PRODUCT IDENTIFICATION SYSTEM

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.

| <u>PART NO.</u>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | <u>[X]<sup>(2)</sup></u> | <u>-</u> | <u>X</u>          | <u>/XX</u> | <u>XXX</u> |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------|----------|-------------------|------------|------------|
| Device                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Tape and Reel Option     |          | Temperature Range | Package    | Pattern    |
| <p><b>Device:</b> PIC18F25K83, PIC18LF25K83<br/>PIC18F26K83, PIC18LF26K83</p> <p><b>Tape and Reel Option:</b> Blank = standard packaging (tube or tray)<br/>T = Tape and Reel<sup>(1), (2)</sup></p> <p><b>Temperature Range:</b> E = -40°C to +125°C (Extended)<br/>I = -40°C to +85°C (Industrial)</p> <p><b>Package:</b> ML = 28-lead QFN 6x6mm<br/>MV = 28-lead UQFN 4x4x0.5mm<br/>SO = 28-lead SOIC<br/>SP = 28-lead Skinny Plastic DIP<br/>SS = 28-lead SSOP</p> <p><b>Pattern:</b> QTP, SQTP, Code or Special Requirements<br/>(blank otherwise)</p>       |                          |          |                   |            |            |
| <p><b>Examples:</b></p> <p>a) PIC18F25K83-E/P 301 = Extended temp., SPDIP package, QTP pattern #301.</p> <p>b) PIC18F26K83-E/SO = Extended temp., SOIC package.</p> <p>c) PIC18F25K83T-I/ML = Tape and reel, Industrial temp., QFN package.</p> <p><b>Note 1:</b> Tape and Reel option is available for ML, MV, SO and SS packages with industrial Temperature Range only.</p> <p><b>2:</b> Tape and Reel identifier only appears in catalog part number description. This identifier is used for ordering purposes and is not printed on the device package.</p> |                          |          |                   |            |            |

---

**Note the following details of the code protection feature on Microchip devices:**

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

---

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

#### **Trademarks**

The Microchip name and logo, the Microchip logo, Adaptec, AnyRate, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Klear, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PackTime, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TempTrackr, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

APT, ClockWorks, The Embedded Control Solutions Company, EtherSynch, FlashTec, Hyper Speed Control, HyperLight Load, IntelliMOS, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet-Wire, SmartFusion, SyncWorld, Temux, TimeCesium, TimeHub, TimePictra, TimeProvider, Vite, WinPath, and ZL are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, BlueSky, BodyCom, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming, ICSP, INICnet, Inter-Chip Connectivity, JitterBlocker, KlearNet, KlearNet logo, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICKit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, SAM-ICE, Serial Quad I/O, SMART-I.S., SQT, SuperSwitcher, SuperSwitcher II, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

The Adaptec logo, Frequency on Demand, Silicon Storage Technology, and Symmcom are registered trademarks of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

© 2017-2020, Microchip Technology Incorporated, All Rights Reserved.

ISBN: 978-1-5224-6389-4

For information regarding Microchip's Quality Management Systems, please visit [www.microchip.com/quality](http://www.microchip.com/quality).

## Worldwide Sales and Service

### AMERICAS

**Corporate Office**  
2355 West Chandler Blvd.  
Chandler, AZ 85224-6199  
Tel: 480-792-7200  
Fax: 480-792-7277  
Technical Support:  
<http://www.microchip.com/support>  
Web Address:  
[www.microchip.com](http://www.microchip.com)

**Atlanta**  
Duluth, GA  
Tel: 678-957-9614  
Fax: 678-957-1455

**Austin, TX**  
Tel: 512-257-3370

**Boston**  
Westborough, MA  
Tel: 774-760-0087  
Fax: 774-760-0088

**Chicago**  
Itasca, IL  
Tel: 630-285-0071  
Fax: 630-285-0075

**Dallas**  
Addison, TX  
Tel: 972-818-7423  
Fax: 972-818-2924

**Detroit**  
Novi, MI  
Tel: 248-848-4000

**Houston, TX**  
Tel: 281-894-5983

**Indianapolis**  
Noblesville, IN  
Tel: 317-773-8323  
Fax: 317-773-5453  
Tel: 317-536-2380

**Los Angeles**  
Mission Viejo, CA  
Tel: 949-462-9523  
Fax: 949-462-9608  
Tel: 951-273-7800

**Raleigh, NC**  
Tel: 919-844-7510

**New York, NY**  
Tel: 631-435-6000

**San Jose, CA**  
Tel: 408-735-9110  
Tel: 408-436-4270

**Canada - Toronto**  
Tel: 905-695-1980  
Fax: 905-695-2078

### ASIA/PACIFIC

**Australia - Sydney**  
Tel: 61-2-9868-6733

**China - Beijing**  
Tel: 86-10-8569-7000

**China - Chengdu**  
Tel: 86-28-8665-5511

**China - Chongqing**  
Tel: 86-23-8980-9588

**China - Dongguan**  
Tel: 86-769-8702-9880

**China - Guangzhou**  
Tel: 86-20-8755-8029

**China - Hangzhou**  
Tel: 86-571-8792-8115

**China - Hong Kong SAR**  
Tel: 852-2943-5100

**China - Nanjing**  
Tel: 86-25-8473-2460

**China - Qingdao**  
Tel: 86-532-8502-7355

**China - Shanghai**  
Tel: 86-21-3326-8000

**China - Shenyang**  
Tel: 86-24-2334-2829

**China - Shenzhen**  
Tel: 86-755-8864-2200

**China - Suzhou**  
Tel: 86-186-6233-1526

**China - Wuhan**  
Tel: 86-27-5980-5300

**China - Xian**  
Tel: 86-29-8833-7252

**China - Xiamen**  
Tel: 86-592-2388138

**China - Zhuhai**  
Tel: 86-756-3210040

### ASIA/PACIFIC

**India - Bangalore**  
Tel: 91-80-3090-4444

**India - New Delhi**  
Tel: 91-11-4160-8631

**India - Pune**  
Tel: 91-20-4121-0141

**Japan - Osaka**  
Tel: 81-6-6152-7160

**Japan - Tokyo**  
Tel: 81-3-6880-3770

**Korea - Daegu**  
Tel: 82-53-744-4301

**Korea - Seoul**  
Tel: 82-2-554-7200

**Malaysia - Kuala Lumpur**  
Tel: 60-3-7651-7906

**Malaysia - Penang**  
Tel: 60-4-227-8870

**Philippines - Manila**  
Tel: 63-2-634-9065

**Singapore**  
Tel: 65-6334-8870

**Taiwan - Hsin Chu**  
Tel: 886-3-577-8366

**Taiwan - Kaohsiung**  
Tel: 886-7-213-7830

**Taiwan - Taipei**  
Tel: 886-2-2508-8600

**Thailand - Bangkok**  
Tel: 66-2-694-1351

**Vietnam - Ho Chi Minh**  
Tel: 84-28-5448-2100

### EUROPE

**Austria - Wels**  
Tel: 43-7242-2244-39  
Fax: 43-7242-2244-393

**Denmark - Copenhagen**  
Tel: 45-4485-5910  
Fax: 45-4485-2829

**Finland - Espoo**  
Tel: 358-9-4520-820

**France - Paris**  
Tel: 33-1-69-53-63-20  
Fax: 33-1-69-30-90-79

**Germany - Garching**  
Tel: 49-8931-9700

**Germany - Haan**  
Tel: 49-2129-3766400

**Germany - Heilbronn**  
Tel: 49-7131-72400

**Germany - Karlsruhe**  
Tel: 49-721-625370

**Germany - Munich**  
Tel: 49-89-627-144-0  
Fax: 49-89-627-144-44

**Germany - Rosenheim**  
Tel: 49-8031-354-560

**Israel - Ra'anana**  
Tel: 972-9-744-7705

**Italy - Milan**  
Tel: 39-0331-742611  
Fax: 39-0331-466781

**Italy - Padova**  
Tel: 39-049-7625286

**Netherlands - Drunen**  
Tel: 31-416-690399  
Fax: 31-416-690340

**Norway - Trondheim**  
Tel: 47-7288-4388

**Poland - Warsaw**  
Tel: 48-22-3325737

**Romania - Bucharest**  
Tel: 40-21-407-87-50

**Spain - Madrid**  
Tel: 34-91-708-08-90  
Fax: 34-91-708-08-91

**Sweden - Gothenberg**  
Tel: 46-31-704-60-40

**Sweden - Stockholm**  
Tel: 46-8-5090-4654

**UK - Wokingham**  
Tel: 44-118-921-5800  
Fax: 44-118-921-5820

## Looking for pricing, stock, or lifecycle information?

Click below to explore more details on WIN SOURCE:

 [View PIC18F25K83-E/SS](#) on WIN SOURCE

 [Microchip Technology](#) Information

## Optimize Your Supply Chain with WIN SOURCE Solutions

-  Global Sourcing Solution
-  Obsolete Management
-  Cost Control Management
-  Shortage Management
-  Alternative Solution
-  Excess Inventory Management