28/40/44/48-Pin, Low-Power High-Performance Microcontrollers with XLP Technology

Description

The PIC18(L)F26/27/45/46/47/55/56/57K42 microcontrollers are available in 28/40/44/48-pin devices. These devices feature a 12-bit ADC with Computation (ADC²) automating Capacitive Voltage Divider (CVD) techniques for advanced touch sensing, averaging, filtering, oversampling and threshold comparison, Temperature Sensor, Vectored Interrupt Controller with fixed latency for handling interrupts, System Bus Arbiter, Direct Memory Access capabilities, UART with support for Asynchronous, DMX, DALI and LIN transmissions, SPI, I²C, memory features like Memory Access Partition (MAP) to support customers in data protection and bootloader applications, and Device Information Area (DIA) which stores factory calibration values to help improve temperature sensor accuracy.

Core Features

- **C Compiler Optimized RISC Architecture**
- **Operating Speed:**
  - Up to 64 MHz clock input
  - 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 128 KB Flash Program Memory
- Up to 8 KB Data SRAM Memory
- Up to 1 KB Data EEPROM Memory
- 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 (PIC18LF26/27/45/46/55/56/57K42)
  - 2.3V to 5.5V (PIC18F26/27/45/46/47/55/56/57K42)
- **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:
  - 5 uA @ 32 kHz, 1.8V, typical
  - 65 uA/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
  - One of the UART modules supports LIN Master and Slave, DMX-512 mode, DALI Gear and Device protocols
  - Automatic and user-timed BREAK period generation
  - DMA Compatible
  - Automatic checksums
  - Programmable 1, 1.5, and 2 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
- Two I²C modules, SMBus, PMBus™ compatible:
  - Supports Standard-mode (100 kHz), Fast-mode (400 kHz) and Fast-mode plus (1 MHz) modes of operation
  - 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
  - I²C, SMBus 2.0 and SMBus 3.0, and 1.8V input level selections
- Device I/O Port Features:
  - 24 I/O pins (PIC18(L)F2xK42)
  - 35 I/O pins (PIC18(L)F4xK42)
  - 43 I/O pins (PIC18(L)F5xK42)
  - One input-only pin (RE3)
  - Individually programmable I/O direction, open-drain, slew rate, weak pull-up control
  - Interrupt-on-change (on up to 25 I/O pins)
  - Three External Interrupt Pins
- Peripheral Pin Select (PPS):
  - Enables pin mapping of digital I/O
- Signal Measurement Timer (SMT):
  - 24-bit timer/counter with prescaler
Analog Peripherals

• Analog-to-Digital Converter with Computation (ADC²):
  - 12-bit with up to 35 external channels
  - 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
  - ±1% at calibration (nominal)

• Low-Power Internal 32 kHz Oscillator (LFINTOSC)

• External 32 kHz Crystal Oscillator (SOSC)

• 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)F2X/4X/5XK42 FAMILY TYPES

<table>
<thead>
<tr>
<th>Device</th>
<th>Data Sheet Index</th>
<th>Program Flash Memory (KB)</th>
<th>Data EEPROM (B)</th>
<th>Data SRAM (bytes)</th>
<th>I/O Pins</th>
<th>12-bit ADC</th>
<th>5-bit DAC</th>
<th>Comparator</th>
<th>Window Watchdog Timer (WWDt)</th>
<th>Signal Measurement Timer (SMT)</th>
<th>8-bit/16-bit Timer (with HLT)</th>
<th>CCP/10-bit PWM</th>
<th>CWG</th>
<th>NCO</th>
<th>CLC</th>
<th>Zero-Cross Detect</th>
<th>Direct Memory Access (DMA)</th>
<th>Memory Access Partition</th>
<th>Vectored Interrupts</th>
<th>UART</th>
<th>ICSP</th>
<th>Peripheral Pin Select</th>
<th>Peripheral Module Disable</th>
<th>Debug</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIC18(L)F24K42</td>
<td>A</td>
<td>16</td>
<td>256</td>
<td>1024</td>
<td>25</td>
<td>24</td>
<td>1</td>
<td>2</td>
<td>3/4</td>
<td>Y</td>
<td>Y</td>
<td>4/4</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>Y</td>
<td>2</td>
<td>Y</td>
<td>Y</td>
<td>2</td>
<td>2/1</td>
<td>Y</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td>PIC18(L)F25K42</td>
<td>A</td>
<td>32</td>
<td>256</td>
<td>2048</td>
<td>25</td>
<td>24</td>
<td>1</td>
<td>2</td>
<td>3/4</td>
<td>Y</td>
<td>Y</td>
<td>4/4</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>Y</td>
<td>2</td>
<td>Y</td>
<td>Y</td>
<td>2</td>
<td>2/1</td>
<td>Y</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td>PIC18(L)F26K42</td>
<td>B</td>
<td>64</td>
<td>1024</td>
<td>4096</td>
<td>25</td>
<td>24</td>
<td>1</td>
<td>2</td>
<td>3/4</td>
<td>Y</td>
<td>Y</td>
<td>4/4</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>Y</td>
<td>2</td>
<td>Y</td>
<td>Y</td>
<td>2</td>
<td>2/1</td>
<td>Y</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td>PIC18(L)F27K42</td>
<td>B</td>
<td>128</td>
<td>1024</td>
<td>8192</td>
<td>25</td>
<td>24</td>
<td>1</td>
<td>2</td>
<td>3/4</td>
<td>Y</td>
<td>Y</td>
<td>4/4</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>Y</td>
<td>2</td>
<td>Y</td>
<td>Y</td>
<td>2</td>
<td>2/1</td>
<td>Y</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td>PIC18(L)F45K42</td>
<td>B</td>
<td>32</td>
<td>256</td>
<td>2048</td>
<td>36</td>
<td>35</td>
<td>1</td>
<td>2</td>
<td>3/4</td>
<td>Y</td>
<td>Y</td>
<td>4/4</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>Y</td>
<td>2</td>
<td>Y</td>
<td>Y</td>
<td>2</td>
<td>2/1</td>
<td>Y</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td>PIC18(L)F46K42</td>
<td>B</td>
<td>64</td>
<td>1024</td>
<td>4096</td>
<td>36</td>
<td>35</td>
<td>1</td>
<td>2</td>
<td>3/4</td>
<td>Y</td>
<td>Y</td>
<td>4/4</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>Y</td>
<td>2</td>
<td>Y</td>
<td>Y</td>
<td>2</td>
<td>2/1</td>
<td>Y</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td>PIC18(L)F47K42</td>
<td>B</td>
<td>128</td>
<td>1024</td>
<td>8192</td>
<td>36</td>
<td>35</td>
<td>1</td>
<td>2</td>
<td>3/4</td>
<td>Y</td>
<td>Y</td>
<td>4/4</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>Y</td>
<td>2</td>
<td>Y</td>
<td>Y</td>
<td>2</td>
<td>2/1</td>
<td>Y</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td>PIC18(L)F55K42</td>
<td>B</td>
<td>32</td>
<td>256</td>
<td>2048</td>
<td>44</td>
<td>43</td>
<td>1</td>
<td>2</td>
<td>3/4</td>
<td>Y</td>
<td>Y</td>
<td>4/4</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>Y</td>
<td>2</td>
<td>Y</td>
<td>Y</td>
<td>2</td>
<td>2/1</td>
<td>Y</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td>PIC18(L)F56K42</td>
<td>B</td>
<td>64</td>
<td>1024</td>
<td>4096</td>
<td>44</td>
<td>43</td>
<td>1</td>
<td>2</td>
<td>3/4</td>
<td>Y</td>
<td>Y</td>
<td>4/4</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>Y</td>
<td>2</td>
<td>Y</td>
<td>Y</td>
<td>2</td>
<td>2/1</td>
<td>Y</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td>PIC18(L)F57K42</td>
<td>B</td>
<td>128</td>
<td>1024</td>
<td>8192</td>
<td>44</td>
<td>43</td>
<td>1</td>
<td>2</td>
<td>3/4</td>
<td>Y</td>
<td>Y</td>
<td>4/4</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>Y</td>
<td>2</td>
<td>Y</td>
<td>Y</td>
<td>2</td>
<td>2/1</td>
<td>Y</td>
<td>I</td>
<td></td>
</tr>
</tbody>
</table>

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

**Data Sheet Index:**

Unshaded devices are not described in this document.

A: DS40001869 PIC18(L)F24/25K42 Data Sheet, 28-Pin
B: DS40001919 PIC18(L)F26/27/45/46/47/55/56/57K42 Data Sheet, 28/40/44/48-Pin

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

28-pin SPDIP, SOIC, SSOP

Note: See Table 1 for location of all peripheral functions.

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

Note 1: See Table 1 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.
Note: See Table 2 for location of all peripheral functions.

Note 1: See Table 2 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.
44-pin QFN (8x8x0.9mm)

Note 1: See Table 2 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.

44-pin TQFP (10x10x1mm)

Note: See Table 2 for location of all peripheral functions.
Note: See Table 3 for location of all peripheral functions.
### Pin Allocation Tables

**TABLE 1: 28-PIN ALLOCATION TABLE (PIC18(L)F2XK42)**

<table>
<thead>
<tr>
<th>I/O</th>
<th>28-Pin SPDIP/SSOP/SSOP</th>
<th>28-Pin (U)QFN</th>
<th>ADC</th>
<th>Voltage Reference</th>
<th>DAC</th>
<th>Comparators</th>
<th>Zero Cross Detect</th>
<th>I²C</th>
<th>SPI</th>
<th>UART</th>
<th>DSM</th>
<th>Timers/SMT</th>
<th>CCP and PWM</th>
<th>CWG</th>
<th>CLC</th>
<th>NCO</th>
<th>Clock Reference (CLKR)</th>
<th>Interrupt-on-Change</th>
<th>Basic</th>
</tr>
</thead>
<tbody>
<tr>
<td>RA0</td>
<td>2</td>
<td>27</td>
<td>AREA</td>
<td>—</td>
<td>—</td>
<td>C1IN0-</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RA1</td>
<td>3</td>
<td>28</td>
<td>AREA</td>
<td>—</td>
<td>—</td>
<td>C1IN1-</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RA2</td>
<td>4</td>
<td>1</td>
<td>AREA</td>
<td>VREF+</td>
<td>—</td>
<td>DAC1OUT1</td>
<td>C1IN0+</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RA3</td>
<td>5</td>
<td>2</td>
<td>AREA</td>
<td>VREF+</td>
<td>—</td>
<td>C1IN1+</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RA4</td>
<td>6</td>
<td>3</td>
<td>AREA</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RA5</td>
<td>7</td>
<td>4</td>
<td>AREA</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RA6</td>
<td>10</td>
<td>7</td>
<td>AREA</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RA7</td>
<td>9</td>
<td>6</td>
<td>AREA</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RB0</td>
<td>21</td>
<td>18</td>
<td>ANB0</td>
<td>—</td>
<td>—</td>
<td>C2IN1+</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RB1</td>
<td>22</td>
<td>19</td>
<td>ANB1</td>
<td>—</td>
<td>—</td>
<td>C1IN3-</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RB2</td>
<td>23</td>
<td>20</td>
<td>ANB2</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RB3</td>
<td>24</td>
<td>21</td>
<td>ANB3</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RB4</td>
<td>25</td>
<td>22</td>
<td>ANB4</td>
<td>ADCACTR(1)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RB5</td>
<td>26</td>
<td>23</td>
<td>ANB5</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RB6</td>
<td>27</td>
<td>24</td>
<td>ANB6</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RB7</td>
<td>28</td>
<td>25</td>
<td>ANB7</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

**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 I²C and SMB™ 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., RA5) will operate, but input logic levels will be standard TTL/ST as selected by the INLVL register, instead of the I²C specific or SMBus input buffer thresholds.
## TABLE 1: 28-PIN ALLOCATION TABLE (PIC18(L)F2XK42) (CONTINUED)

<table>
<thead>
<tr>
<th>I/O</th>
<th>28-Pin SPDIP/SG/SSOP 28-Pin UQFN</th>
<th>28-Pin (U)QFN</th>
<th>ADC</th>
<th>DAC</th>
<th>Comparators</th>
<th>Zero Cross Detect</th>
<th>I²C</th>
<th>SPI</th>
<th>UART</th>
<th>DSM</th>
<th>Timers/SMT</th>
<th>CCP and PWM</th>
<th>CWG</th>
<th>CLC</th>
<th>NCO</th>
<th>Clock Reference (CLKR)</th>
<th>Interrupt-on-Change</th>
<th>Basic</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC0</td>
<td>11</td>
<td>8</td>
<td>ANC0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>T1CK(1)</td>
<td>T3CK(1)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RC1</td>
<td>12</td>
<td>9</td>
<td>ANC1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>SMTSIG1(1)</td>
<td>CCP2(1)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>IOCC1</td>
<td>SOSC1</td>
</tr>
<tr>
<td>RC2</td>
<td>13</td>
<td>10</td>
<td>ANC2</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>T5CK(1)</td>
<td>CCP3(1)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>IOCC2</td>
<td>—</td>
</tr>
<tr>
<td>RC3</td>
<td>14</td>
<td>11</td>
<td>ANC3</td>
<td>—</td>
<td>—</td>
<td>SCL(1,4)</td>
<td>SCK1(1)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>T2I(1)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>IOCC3</td>
<td>—</td>
</tr>
<tr>
<td>RC4</td>
<td>15</td>
<td>12</td>
<td>ANC4</td>
<td>—</td>
<td>—</td>
<td>SDA1(1,4)</td>
<td>SDI1(1)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>T4I(1)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>IOCC4</td>
<td>—</td>
</tr>
<tr>
<td>RC5</td>
<td>16</td>
<td>13</td>
<td>ANC5</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>T4I(1)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>IOCC5</td>
<td>—</td>
</tr>
<tr>
<td>RC6</td>
<td>17</td>
<td>14</td>
<td>ANC6</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CTS1(1)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>IOCC6</td>
<td>—</td>
</tr>
<tr>
<td>RC7</td>
<td>18</td>
<td>15</td>
<td>ANC7</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Rxx(1)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>IOCC7</td>
<td>—</td>
</tr>
<tr>
<td>RE3</td>
<td>26</td>
<td>1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>IOCE3</td>
<td>MCLR</td>
</tr>
<tr>
<td>Vcc</td>
<td>20</td>
<td>17</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Vss</td>
<td>8, 9</td>
<td>5, 16</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>OUT(2)</td>
<td>—</td>
<td>ADGRDA</td>
<td>—</td>
<td>—</td>
<td>ADGRDB</td>
<td>C1OUT</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>DSM</td>
<td>TMR0</td>
<td>CCP1</td>
<td>CCP2</td>
<td>CCP3</td>
<td>PWMSO</td>
<td>CLC1OUT</td>
<td>—</td>
</tr>
</tbody>
</table>

**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 I²C and SMB™ 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., RA5) will operate, but input logic levels will be standard TTL/ST as selected by the INLVL register, instead of the PC specific or SMBus input buffer thresholds.
### TABLE 2: 40/44-PIN ALLOCATION TABLE FOR PIC18(L)F4XK42

<table>
<thead>
<tr>
<th>I/O</th>
<th>40-Pin PDIP</th>
<th>44-Pin TQFP</th>
<th>40-Pin UCQFN</th>
<th>44-Pin QFN</th>
<th>ADC</th>
<th>Voltage Reference</th>
<th>DAC</th>
<th>Comparators</th>
<th>Zero Cross Detect</th>
<th>I²C</th>
<th>SPI</th>
<th>UART</th>
<th>DSM</th>
<th>Timers/SMT</th>
<th>CCP and PWM</th>
<th>CWG</th>
<th>CLC</th>
<th>NCO</th>
<th>Clock Reference (CLKR)</th>
<th>Interrupt-on-Change</th>
<th>Basic</th>
</tr>
</thead>
<tbody>
<tr>
<td>RA0</td>
<td>2</td>
<td>19</td>
<td>17</td>
<td>19</td>
<td>ANA0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RA1</td>
<td>3</td>
<td>20</td>
<td>18</td>
<td>20</td>
<td>ANA1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RA2</td>
<td>4</td>
<td>21</td>
<td>19</td>
<td>21</td>
<td>ANA2</td>
<td>VREF-</td>
<td>DAC1OUT</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RA3</td>
<td>5</td>
<td>22</td>
<td>20</td>
<td>22</td>
<td>ANA3</td>
<td>VREF+</td>
<td>—</td>
<td>C1IN1+</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RA4</td>
<td>6</td>
<td>23</td>
<td>21</td>
<td>23</td>
<td>ANA4</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RA5</td>
<td>7</td>
<td>24</td>
<td>22</td>
<td>24</td>
<td>ANA5</td>
<td>—</td>
<td>—</td>
<td>SS1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RA6</td>
<td>13</td>
<td>31</td>
<td>29</td>
<td>33</td>
<td>ANA6</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RA7</td>
<td>30</td>
<td>28</td>
<td>32</td>
<td>30</td>
<td>ANA7</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RB0</td>
<td>33</td>
<td>8</td>
<td>8</td>
<td>9</td>
<td>ANB0</td>
<td>—</td>
<td>—</td>
<td>C2IN1+</td>
<td>ZCD</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RB1</td>
<td>34</td>
<td>9</td>
<td>9</td>
<td>10</td>
<td>ANB1</td>
<td>—</td>
<td>—</td>
<td>C1IN3-</td>
<td>C2IN3-</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RB2</td>
<td>35</td>
<td>10</td>
<td>10</td>
<td>11</td>
<td>ANB2</td>
<td>—</td>
<td>—</td>
<td>SDA2I(4)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RB3</td>
<td>36</td>
<td>11</td>
<td>11</td>
<td>12</td>
<td>ANB3</td>
<td>—</td>
<td>—</td>
<td>C1IN2-</td>
<td>C2IN2-</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RB4</td>
<td>37</td>
<td>12</td>
<td>14</td>
<td>14</td>
<td>ANB4</td>
<td>—</td>
<td>—</td>
<td>ADCACIIT(1)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RB5</td>
<td>38</td>
<td>15</td>
<td>13</td>
<td>15</td>
<td>ANB5</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RB6</td>
<td>39</td>
<td>16</td>
<td>14</td>
<td>16</td>
<td>ANB6</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RB7</td>
<td>40</td>
<td>17</td>
<td>15</td>
<td>17</td>
<td>ANB7</td>
<td>—</td>
<td>DAC1OUT</td>
<td>—</td>
<td>RX2(1)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RC0</td>
<td>15</td>
<td>32</td>
<td>30</td>
<td>34</td>
<td>ANC0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RC1</td>
<td>16</td>
<td>35</td>
<td>31</td>
<td>35</td>
<td>ANC1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RC2</td>
<td>17</td>
<td>36</td>
<td>32</td>
<td>36</td>
<td>ANC2</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

**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 I²C and SMB™ 3.0/2.0 logic levels; The SCLx/SDAx signals may be assigned to any of the RB1/RB2/RC3/RC4/RD0/RD1 pins. PPS assignments to the other pins (e.g., RA5) will operate, but input logic levels will be standard TTL/ST as selected by the INLVL register, instead of the I²C specific or SMBus input buffer thresholds.
## TABLE 2: 40/44-PIN ALLOCATION TABLE FOR PIC18(L)F4XK42

<table>
<thead>
<tr>
<th>IO</th>
<th>40-Pin PDIP</th>
<th>44-Pin TQFP</th>
<th>40-Pin UQFN</th>
<th>44-Pin QFN</th>
<th>ADC</th>
<th>Voltage Reference</th>
<th>DAC</th>
<th>Comparators</th>
<th>Zero Cross Detect</th>
<th>I²C</th>
<th>SPI</th>
<th>UART</th>
<th>DSM</th>
<th>Timers/SMT</th>
<th>CCP and PWM</th>
<th>CWG</th>
<th>CLC</th>
<th>NCO</th>
<th>Clock Reference (CLKR)</th>
<th>Interrupt-on-Change</th>
<th>Basic</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC3</td>
<td>18</td>
<td>37</td>
<td>33</td>
<td>37</td>
<td>ANC3</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>SCL1(3,4)</td>
<td>SCK1(1)</td>
<td>—</td>
<td>—</td>
<td>T2IN(1)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>IOCC3</td>
</tr>
<tr>
<td>RC4</td>
<td>23</td>
<td>42</td>
<td>38</td>
<td>42</td>
<td>ANC4</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>SDA1(3,4)</td>
<td>SDI1(1)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>IOCC4</td>
</tr>
<tr>
<td>RC5</td>
<td>24</td>
<td>43</td>
<td>39</td>
<td>43</td>
<td>ANC5</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>T4IN(1)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>IOCC5</td>
</tr>
<tr>
<td>RC6</td>
<td>25</td>
<td>44</td>
<td>40</td>
<td>44</td>
<td>ANC6</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>IOCC6</td>
</tr>
<tr>
<td>RC7</td>
<td>26</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>ANC7</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>RX1(1)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>IOCC7</td>
</tr>
<tr>
<td>RD0</td>
<td>19</td>
<td>38</td>
<td>34</td>
<td>38</td>
<td>AND0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RD1</td>
<td>20</td>
<td>39</td>
<td>35</td>
<td>39</td>
<td>AND1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RD2</td>
<td>21</td>
<td>40</td>
<td>36</td>
<td>40</td>
<td>AND2</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RD3</td>
<td>22</td>
<td>41</td>
<td>37</td>
<td>41</td>
<td>AND3</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RD4</td>
<td>27</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>AND4</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RD5</td>
<td>28</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>AND5</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RD6</td>
<td>29</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>AND6</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RD7</td>
<td>30</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>AND7</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RE0</td>
<td>8</td>
<td>25</td>
<td>23</td>
<td>25</td>
<td>ANE0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RE1</td>
<td>9</td>
<td>26</td>
<td>24</td>
<td>26</td>
<td>ANE1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RE2</td>
<td>10</td>
<td>27</td>
<td>25</td>
<td>27</td>
<td>ANE2</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RE3</td>
<td>1</td>
<td>18</td>
<td>16</td>
<td>18</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RE0</td>
<td>11</td>
<td>32</td>
<td>28</td>
<td>32</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RE1</td>
<td>12</td>
<td>31</td>
<td>29</td>
<td>31</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

**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 I²C and SMB™ 3.0/2.0 logic levels. The SCLx/SDAx signals may be assigned to any of the RB1/RB2/RC3/RC4/RD0/RD1 pins. PPS assignments to the other pins (e.g., RA5) will operate, but input logic levels will be standard TTL/ST as selected by the INLVL register, instead of the I²C specific or SMBus input buffer thresholds.
### TABLE 2: 40/44-PIN ALLOCATION TABLE FOR PIC18(L)F4XK42

<table>
<thead>
<tr>
<th>I/O</th>
<th>40-Pin PDIP</th>
<th>44-Pin TQFP</th>
<th>40-Pin UQFN</th>
<th>44-Pin QFN</th>
<th>ADC</th>
<th>DAC</th>
<th>Voltage Reference</th>
<th>DAC</th>
<th>Comparators</th>
<th>Zero Cross Detect</th>
<th>SPI</th>
<th>UART</th>
<th>DSM</th>
<th>Timers/SMT</th>
<th>CCP and PWM</th>
<th>CWG</th>
<th>CLC</th>
<th>NCO</th>
<th>Clock Reference (CLKR)</th>
<th>Interrupt-on-Change</th>
<th>Basic</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUT1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>C1OUT</td>
<td>C2OUT</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>ADGRDA</td>
<td>ADGRDB</td>
<td>ADGRDA</td>
<td>ADGRDB</td>
<td>ADGRDB</td>
<td>—</td>
<td>SDA1</td>
<td>SCL1</td>
<td>SDA2</td>
<td>SCL2</td>
<td>SS1</td>
<td>SCK1</td>
<td>SO1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>DTR1</td>
<td>RTS1</td>
<td>TX1</td>
<td>TX2</td>
<td>TX2</td>
<td>DTR2</td>
<td>RTS2</td>
<td>RTS2</td>
<td>TX2</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>DSM</td>
<td>TMR0</td>
<td>CCP1</td>
<td>CCP2</td>
<td>CCP3</td>
<td>CCP4</td>
<td>PWM5OUT</td>
<td>PWM6OUT</td>
<td>PWM7OUT</td>
<td>PWM8OUT</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>CWG</td>
<td>CWG1A</td>
<td>CWG1B</td>
<td>CWG1C</td>
<td>CWG1D</td>
<td>CWG2A</td>
<td>CWG2B</td>
<td>CWG2C</td>
<td>CWG2D</td>
<td>CWG3A</td>
<td>CWG3B</td>
<td>CWG3C</td>
<td>CWG3D</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>CLC</td>
<td>CLC1OUT</td>
<td>CLC2OUT</td>
<td>CLC3OUT</td>
<td>CLC4OUT</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>NCO</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>CLKR</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

**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 I2C and SMB™ 3.0/2.0 logic levels. The SCLx/SDAx signals may be assigned to any of the R81/R82/R83/RC1/RC2/RC4/RC5/RC6/RC1 pins. PPS assignments to the other pins (e.g., RA5) will operate, but input logic levels will be standard TTL/ST as selected by the INLVL register, instead of the I2C specific or SMBus input buffer thresholds.
<table>
<thead>
<tr>
<th>I/O</th>
<th>48-Pin TQFP</th>
<th>48-Pin UQFN</th>
<th>ADC</th>
<th>DAC</th>
<th>Comparators</th>
<th>Zero Cross Detect</th>
<th>I²C</th>
<th>SPI</th>
<th>UART</th>
<th>DSM</th>
<th>Timers/SMT</th>
<th>CoP and PWN</th>
<th>CWG</th>
<th>CLC</th>
<th>NCO</th>
<th>Clock Reference (CLKR)</th>
<th>Interrupt-on-Change</th>
<th>Basic</th>
</tr>
</thead>
<tbody>
<tr>
<td>RA0</td>
<td>21</td>
<td>21</td>
<td>ANA0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>IOCA0</td>
<td>—</td>
</tr>
<tr>
<td>RA1</td>
<td>22</td>
<td>22</td>
<td>ANA1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>IOCA1</td>
<td>—</td>
</tr>
<tr>
<td>RA2</td>
<td>23</td>
<td>23</td>
<td>ANA2</td>
<td>VREF</td>
<td>DAC1OUT1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>IOCA2</td>
<td>—</td>
</tr>
<tr>
<td>RA3</td>
<td>24</td>
<td>24</td>
<td>ANA3</td>
<td>VREF</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>IOCA3</td>
<td>—</td>
</tr>
<tr>
<td>RA4</td>
<td>25</td>
<td>25</td>
<td>ANA4</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>IOCA4</td>
<td>—</td>
</tr>
<tr>
<td>RA5</td>
<td>26</td>
<td>26</td>
<td>ANA5</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>IOCA5</td>
<td>—</td>
</tr>
<tr>
<td>RA6</td>
<td>33</td>
<td>33</td>
<td>ANA6</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>IOCA6</td>
<td>OSC2</td>
</tr>
<tr>
<td>RA7</td>
<td>32</td>
<td>32</td>
<td>ANA7</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>IOCA7</td>
<td>OSC1</td>
</tr>
<tr>
<td>RB0</td>
<td>8</td>
<td>8</td>
<td>ANB0</td>
<td>—</td>
<td>C2IN1</td>
<td>ZCD</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>INT0</td>
<td>—</td>
</tr>
<tr>
<td>RB1</td>
<td>9</td>
<td>9</td>
<td>ANB1</td>
<td>—</td>
<td>C1IN3</td>
<td>SCL&lt;sup&gt;2,4&lt;/sup&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>INT1</td>
<td>—</td>
</tr>
<tr>
<td>RB2</td>
<td>10</td>
<td>10</td>
<td>ANB2</td>
<td>—</td>
<td>—</td>
<td>SDAX&lt;sup&gt;2,4&lt;/sup&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>INT2</td>
<td>—</td>
</tr>
<tr>
<td>RB3</td>
<td>11</td>
<td>11</td>
<td>ANB3</td>
<td>—</td>
<td>C1IN2</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>INT3</td>
<td>—</td>
</tr>
<tr>
<td>RB4</td>
<td>16</td>
<td>16</td>
<td>ANB4</td>
<td>ADCACT</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>IOCB3</td>
<td>—</td>
</tr>
<tr>
<td>RB5</td>
<td>17</td>
<td>17</td>
<td>ANB5</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>T1G</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>IOCB4</td>
<td>—</td>
</tr>
<tr>
<td>RB6</td>
<td>18</td>
<td>18</td>
<td>ANB6</td>
<td>—</td>
<td>DAC1OUT2</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>IOCB5</td>
<td>—</td>
</tr>
<tr>
<td>RB7</td>
<td>19</td>
<td>19</td>
<td>ANB7</td>
<td>—</td>
<td>RX2&lt;sup&gt;(1)&lt;/sup&gt;</td>
<td>—</td>
<td>T6IN</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>IOCB6</td>
<td>ICSPCLK</td>
</tr>
<tr>
<td>RC0</td>
<td>34</td>
<td>34</td>
<td>ANC0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>T1CK&lt;sup&gt;(1)&lt;/sup&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>IOCB7</td>
<td>ICSPDAT</td>
</tr>
<tr>
<td>RC1</td>
<td>35</td>
<td>35</td>
<td>ANC1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>SMTSIGN&lt;sup&gt;(1)&lt;/sup&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>IOCB8</td>
<td>SOSC</td>
</tr>
</tbody>
</table>

**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 I²C and SMB™ 3.0/2.0 logic levels. The SCLx/SDAx signals may be assigned to any of the RB1/RB2/RC3/RC4/RD0/RD1 pins. PPS assignments to the other pins (e.g., RA5) will operate, but input logic levels will be standard TTL/ST as selected by the INLVL register, instead of the I²C specific or SMBus input buffer thresholds.
### TABLE 3: 48-PIN ALLOCATION TABLE FOR PIC18(L)F5XK42 (CONTINUED)

<table>
<thead>
<tr>
<th>I/O</th>
<th>48-Pin TQFP</th>
<th>48-Pin UQFN</th>
<th>ADC</th>
<th>DAC</th>
<th>Voltage Reference</th>
<th>Comparators</th>
<th>Zero Cross Detect</th>
<th>I²C</th>
<th>SPI</th>
<th>UART</th>
<th>DSM</th>
<th>Timers/SMT</th>
<th>CCP and PWM</th>
<th>CWG</th>
<th>CLC</th>
<th>NCO</th>
<th>Clock Reference (CLKR)</th>
<th>Interrupt-on-Change</th>
<th>Basic</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC2</td>
<td>40</td>
<td>40</td>
<td>ANC2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RC3</td>
<td>41</td>
<td>41</td>
<td>ANC3</td>
<td></td>
<td></td>
<td>SCL1(3,4)</td>
<td>SCK1(5)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>T2IN(1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RC4</td>
<td>46</td>
<td>46</td>
<td>ANC4</td>
<td></td>
<td></td>
<td>SDA1(3,4)</td>
<td>SDO1(5)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RC5</td>
<td>47</td>
<td>47</td>
<td>ANC5</td>
<td></td>
<td></td>
<td></td>
<td>T4IN(1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RC6</td>
<td>48</td>
<td>48</td>
<td>ANC6</td>
<td></td>
<td></td>
<td>CTS1(1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RC7</td>
<td>1</td>
<td>1</td>
<td>ANC7</td>
<td></td>
<td></td>
<td>RX(1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RD0</td>
<td>42</td>
<td>42</td>
<td>AND0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RD1</td>
<td>43</td>
<td>43</td>
<td>AND1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RD2</td>
<td>44</td>
<td>44</td>
<td>AND2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RD3</td>
<td>45</td>
<td>45</td>
<td>AND3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RD4</td>
<td>2</td>
<td>2</td>
<td>AND4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RD5</td>
<td>3</td>
<td>3</td>
<td>AND5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RD6</td>
<td>4</td>
<td>4</td>
<td>AND6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RD7</td>
<td>5</td>
<td>5</td>
<td>AND7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RE0</td>
<td>27</td>
<td>27</td>
<td>ANE0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RE1</td>
<td>28</td>
<td>28</td>
<td>ANE1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RE2</td>
<td>29</td>
<td>29</td>
<td>ANE2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RE3</td>
<td>20</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RF0</td>
<td>36</td>
<td>36</td>
<td>ANF0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RF1</td>
<td>37</td>
<td>37</td>
<td>ANF1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RF2</td>
<td>38</td>
<td>38</td>
<td>ANF2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RF3</td>
<td>39</td>
<td>39</td>
<td>ANF3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RF4</td>
<td>12</td>
<td>12</td>
<td>ANF4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RF5</td>
<td>13</td>
<td>13</td>
<td>ANF5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RF6</td>
<td>14</td>
<td>14</td>
<td>ANF6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RF7</td>
<td>15</td>
<td>15</td>
<td>ANF7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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 I²C and SMB™ 3.0/2.0 logic levels. The SCLx/SDAx signals may be assigned to any of the RB1/RB2/RC3/RC4/RD0/RD1 pins. PPS assignments to the other pins (e.g., RA5) will operate, but input logic levels will be standard TTL/ST as selected by the INLVL register, instead of the I²C specific or SMBus input buffer thresholds.
### TABLE 3: 48-PIN ALLOCATION TABLE FOR PIC18(L)F5XK42 (CONTINUED)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>VDD</td>
<td>7, 30</td>
<td>7, 30</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>VSS</td>
<td>6, 31</td>
<td>6, 31</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>OUT(1)</td>
<td>—</td>
<td>ADGRDA</td>
<td>ADGRDB</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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^2^C and SMB™ 3.0/2.0 logic levels. The SCLx/SDAx signals may be assigned to any of the RB1/RB2/RC3/RC4/RD0/RD1 pins. PPS assignments to the other pins (e.g., RA5) will operate, but input logic levels will be standard TTL/ST as selected by the INLVL register, instead of the I^2^C specific or SMBus input buffer thresholds.
# Table of Contents

1.0 Device Overview ................................................................. 18
2.0 Guidelines for Getting Started with PIC18(L)F26/27/45/46/47/55/56/57K42 Microcontrollers ................................................. 22
3.0 PIC18 CPU ............................................................................. 25
4.0 Memory Organization .......................................................... 32
5.0 Device Configuration ............................................................ 63
6.0 Resets ................................................................................. 79
7.0 Oscillator Module (with Fail-Safe Clock Monitor) ......................... 90
8.0 Reference Clock Output Module .............................................. 109
9.0 Interrupt Controller .............................................................. 113
10.0 Power-Saving Operation Modes ............................................ 168
11.0 Windowed Watchdog Timer (WWDT) ..................................... 176
12.0 8x8 Hardware Multiplier ..................................................... 185
13.0 Nonvolatile Memory (NVM) Control ..................................... 187
14.0 Cyclic Redundancy Check (CRC) Module with Memory Scanner ........................................................................ 211
15.0 Direct Memory Access (DMA) ............................................... 226
16.0 /O Ports ............................................................................... 257
17.0 Peripheral Pin Select (PPS) Module ........................................ 272
18.0 Interrupt-on-Change ............................................................ 283
19.0 Peripheral Module Disable (PMD) .......................................... 287
20.0 Timer0 Module ....................................................................... 296
21.0 Timer1/3/5 Module with Gate Control ........................................ 302
22.0 Timer2/4/6 Module ............................................................... 317
23.0 Capture/Compare/PWM Module ............................................ 339
24.0 Pulse-Width Modulation (PWM) ........................................... 352
25.0 Signal Measurement Timer (SMT) ........................................... 359
26.0 Complementary Waveform Generator (CWG) Module ................... 403
27.0 Configurable Logic Cell (CLC) ............................................... 431
28.0 Numerically Controlled Oscillator (NCO) Module ...................... 446
29.0 Zero-Cross Detection (ZCD) Module ....................................... 456
30.0 Data Signal Modulator (DSM) Module ..................................... 461
31.0 Universal Asynchronous Receiver Transmitter (UART) With Protocol Support ................................................................. 472
32.0 Serial Peripheral Interface (SPI) Module .................................... 510
33.0 I2C Module .......................................................................... 542
34.0 Fixed Voltage Reference (FVR) ................................................. 594
35.0 Temperature Indicator Module ............................................... 596
36.0 Analog-to-Digital Converter with Computation (ADC2) Module ........................................................................ 599
37.0 5-Bit Digital-to-Analog Converter (DAC) Module .......................... 637
38.0 Comparator Module .............................................................. 641
39.0 High/Low-Voltage Detect (HLVD) .......................................... 650
40.0 In-Circuit Serial Programming™ (ICSP™) .................................. 658
41.0 Instruction Set Summary ....................................................... 660
42.0 Register Summary .................................................................. 714
43.0 Development Support .......................................................... 731
44.0 Electrical Specifications ........................................................ 735
45.0 DC and AC Characteristics Graphs and Tables ............................. 766
46.0 Packaging Information ......................................................... 767
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. 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 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:

- PIC18F26K42
- PIC18F27K42
- PIC18F45K42
- PIC18F46K42
- PIC18F47K42
- PIC18F55K42
- PIC18F56K42
- PIC18F57K42
- PIC18LF26K42
- PIC18LF27K42
- PIC18LF45K42
- PIC18LF46K42
- PIC18LF47K42
- PIC18LF55K42
- PIC18LF56K42
- PIC18LF57K42

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 (I2C), Direct Memory Access (DMA), Configurable Logic Cells (CLC), Signal Measurement Timer (SMT), Numerically Controlled Oscillator (NCO), and Analog-to-Digital Converter with Computation (ADC2).

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.

- **I2C Module:** The I2C module provides a synchronous interface between the microcontroller and other I2C-compatible devices using the two-wire I2C serial bus. Devices communicate in a master/slave environment. The I2C 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, reduces code overhead. It has a new module called ADC2 with computation features, which provides a digital filter and threshold interrupt functions.

1.2 Details on Individual Family Members

Devices in the PIC18(L)F26/27/45/46/47/55/56/57K42 family are available in 28-pin and 40/44/48-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)F2X/4X/5XK42 Family Types Table (page 4). The pinouts for all devices are listed in Table 1.
<table>
<thead>
<tr>
<th>Features</th>
<th>PIC18(L)F26K42</th>
<th>PIC18(L)F27K42</th>
<th>PIC18(L)F45K42</th>
<th>PIC18(L)F46K42</th>
<th>PIC18(L)F47K42</th>
<th>PIC18(L)F55K42</th>
<th>PIC18(L)F56K42</th>
<th>PIC18(L)F57K42</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program Memory (Bytes)</td>
<td>65536</td>
<td>131072</td>
<td>32768</td>
<td>65536</td>
<td>131072</td>
<td>32768</td>
<td>655536</td>
<td>131072</td>
</tr>
<tr>
<td>Program Memory (Instructions)</td>
<td>32768</td>
<td>65536</td>
<td>16384</td>
<td>32768</td>
<td>65536</td>
<td>16384</td>
<td>32768</td>
<td>65536</td>
</tr>
<tr>
<td>Data Memory (Bytes)</td>
<td>4096</td>
<td>8192</td>
<td>2048</td>
<td>4096</td>
<td>8192</td>
<td>2048</td>
<td>4096</td>
<td>8192</td>
</tr>
<tr>
<td>Data EEPROM Memory (Bytes)</td>
<td>1024</td>
<td>1024</td>
<td>256</td>
<td>1024</td>
<td>256</td>
<td>1024</td>
<td>1024</td>
<td>1024</td>
</tr>
<tr>
<td></td>
<td>28-pin SPDIP</td>
<td>28-pin SOIC</td>
<td>28-pin SSOP</td>
<td>28-pin QFN</td>
<td>40-pin UQFN</td>
<td>40-pin UQFN</td>
<td>40-pin UQFN</td>
<td>40-pin UQFN</td>
</tr>
<tr>
<td></td>
<td>28-pin SSOP</td>
<td>28-pin SOIC</td>
<td>28-pin SSOP</td>
<td>28-pin QFN</td>
<td>44-pin TQFP</td>
<td>44-pin TQFP</td>
<td>44-pin TQFP</td>
<td>44-pin TQFP</td>
</tr>
<tr>
<td></td>
<td>28-pin QFN</td>
<td>28-pin SSOP</td>
<td>28-pin SOIC</td>
<td>28-pin SSOP</td>
<td>44-pin TQFP</td>
<td>44-pin TQFP</td>
<td>44-pin TQFP</td>
<td>44-pin UQFN</td>
</tr>
<tr>
<td></td>
<td>28-pin QFN</td>
<td>28-pin SSOP</td>
<td>28-pin SOIC</td>
<td>28-pin SSOP</td>
<td>44-pin UQFN</td>
<td>48-pin TQFP</td>
<td>48-pin UQFN</td>
<td>48-pin TQFP</td>
</tr>
<tr>
<td></td>
<td>28-pin UQFN</td>
<td>28-pin SSOP</td>
<td>28-pin SOIC</td>
<td>28-pin SSOP</td>
<td>44-pin UQFN</td>
<td>48-pin UQFN</td>
<td>48-pin UQFN</td>
<td>48-pin UQFN</td>
</tr>
<tr>
<td></td>
<td>28-pin UQFN</td>
<td>28-pin SSOP</td>
<td>28-pin SOIC</td>
<td>28-pin SSOP</td>
<td>44-pin UQFN</td>
<td>48-pin TQFP</td>
<td>48-pin UQFN</td>
<td>48-pin TQFP</td>
</tr>
<tr>
<td>12-Bit Analog-to-Digital</td>
<td>5 internal</td>
<td>5 internal</td>
<td>5 internal</td>
<td>5 internal</td>
<td>5 internal</td>
<td>5 internal</td>
<td>5 internal</td>
<td>5 internal</td>
</tr>
<tr>
<td>Conversion Module (ADC&lt;sup&gt;2&lt;/sup&gt;) with</td>
<td>24 external</td>
<td>24 external</td>
<td>35 external</td>
<td>35 external</td>
<td>43 external</td>
<td>43 external</td>
<td>43 external</td>
<td>43 external</td>
</tr>
<tr>
<td>Computation Accelerator</td>
<td>Capture/Compare/PWM Modules (CCP)</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>10-Bit Pulse-Width Modulator (PWM)</td>
<td>4/3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Timers (16-/8-bit)</td>
<td>1 UART, 1 UART with DMX/DALI/LIN, 2 I&lt;sup&gt;2&lt;/sup&gt;C, 1 SPI</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Serial Communications</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Complementary Waveform</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Generator (CWG)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Zero-Cross Detect (ZCD)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Data Signal Modulator (DSM)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Signal Measurement Timer (SMT)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5-bit Digital to Analog</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Converter (DAC)</td>
<td>Numerically Controlled Oscillator (NCO)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>Features</th>
<th>PIC18(L)F26K42</th>
<th>PIC18(L)F27K42</th>
<th>PIC18(L)F45K42</th>
<th>PIC18(L)F46K42</th>
<th>PIC18(L)F47K42</th>
<th>PIC18(L)F55K42</th>
<th>PIC18(L)F56K42</th>
<th>PIC18(L)F57K42</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comparator Module</td>
<td>2</td>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Direct Memory Access (DMA)</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Configurable Logic Cell (CLC)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Peripheral Pin Select (PPS)</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Peripheral Module Disable (PMD)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16-bit CRC with Scanner</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Programmable High/Low-Voltage Detect (HLVD)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resets (and Delays)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>POR, Programmable BOR,</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\texttt{RESET} Instruction,</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stack Overflow,</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stack Underflow</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(PWRT, OST)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\texttt{MCLR, WDT, MEMV}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Instruction Set</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>81 Instructions;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>87 with Extended Instruction Set enabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum Operating Frequency</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>64 MHz</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\textbf{Note 1:} PORTE is partially implemented. Pin RE3 is an input-only pin on 28/40/44/48-pin variants. In addition to that, on 40/44/48-pin variants, PORTE also consists of RE0, RE1 and RE2 pins.
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 `RegisterName.bits.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 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 name 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)F26/27/45/46/47/55/56/57K42 MICROCONTROLLERS

2.1 Basic Connection Requirements

Getting started with the PIC18(L)F26/27/45/46/47/55/56/57K42 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 μF (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, make sure 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 μF to 0.001 μF. 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 μF in parallel with 0.001 μF).
- 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 μF to 47 μF.
2.3 Master Clear (MCLR) Pin

The 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 MCLR pin. Consequently, specific voltage levels (VIH and VIL) 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 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 MCLR pin should be placed within 0.25 inch (6 mm) of the pin.

FIGURE 2-2: EXAMPLE OF MCLR PIN CONNECTIONS

Note 1: R1 ≤ 10 kΩ is recommended. A suggested starting value is 10 kΩ. Ensure that the MCLR pin VIH and VIL specifications are met.

Note 2: R2 ≤ 470Ω will limit any current flowing into MCLR from the external capacitor, C1, in the event of MCLR pin breakdown, due to Electrostatic Discharge (ESD) or Electrical Overstress (EOS). Ensure that the MCLR pin VIH and VIL specifications are met.

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 (VIH) and input low (VIL) 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 43.0 “Development Support”.
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):

- AN826, “Crystal Oscillator Basics and Crystal Selection for rFIC™ 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)F26/27/45/46/47/55/56/57K42 FAMILY BLOCK DIAGRAM

Note 1: RE3 is only available when MCLR functionality is disabled.

Note 2: OSC1/CLKIN and OSC2/CLKOUT are only available in select oscillator modes and when these pins are not being used as digital I/O. Refer to Section 7.0, Oscillator Module (with Fail-Safe Clock Monitor) for additional information.
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**

<table>
<thead>
<tr>
<th>Selection</th>
<th>Priority register</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Level</td>
<td>ISR</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>MAIN</td>
<td>1</td>
</tr>
<tr>
<td>Peripheral</td>
<td>DMA1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>DMA2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>SCANNER</td>
<td>4</td>
</tr>
</tbody>
</table>
3.1.1 PRIORITY LOCK

The System arbiter grants memory access to the peripheral selections (DMAx, Scanner) when the \texttt{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**

```assembly
; Disable interrupts
BCF INTCON0,GIE

; Bank to PRLOCK register
BANKSEL PRLOCK
MOVFW PRLOCK

; Required sequence, next 4 instructions
MOVLW 55h
MOVWF PRLOCK
MOVFW 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**

```assembly
; Disable interrupts
BCF INTCON0,GIE

; Bank to PRLOCK register
BANKSEL PRLOCK
MOVFW PRLOCK

; Required sequence, next 4 instructions
MOVLW 55h
MOVWF PRLOCK
MOVFW 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 \texttt{MOVLW}, \texttt{CALL}, \texttt{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 \( \rightarrow \) PERIPHERAL PRIORITY \( \rightarrow \) 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 \( \rightarrow \) ISR PRIORITY \( \rightarrow \) MAIN PRIORITY \( \rightarrow \) 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**

<table>
<thead>
<tr>
<th></th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ISPR&lt;2:0&gt;</td>
<td>bit 0</td>
<td></td>
</tr>
</tbody>
</table>

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

- bit 7-3 Unimplemented: Read as ‘0’
- bit 2-0 ISPR<2:0>: Interrupt Service Routine Priority Selection bits

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

<table>
<thead>
<tr>
<th></th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-1/1</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>MAINPR&lt;2:0&gt;</td>
<td>bit 0</td>
<td></td>
</tr>
</tbody>
</table>

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

- 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**

<table>
<thead>
<tr>
<th></th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-1/1</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>DMA1PR&lt;2:0&gt;</td>
<td>bit 0</td>
<td></td>
</tr>
</tbody>
</table>

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

- bit 7-3 Unimplemented: Read as ‘0’
- bit 2-0 DMA1PR<2:0>: DMA1 Priority Selection bits
**PIC18(L)F26/27/45/46/47/55/56/57K42**

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

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DMA2PR&lt;2:0&gt;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as ‘0’
- x = Bit is unknown
- -n/n = Value at POR and BOR/Value at all other Resets
- HS = Hardware set

**Legend:**
- bit 7-3: Unimplemented: Read as ‘0’
- bit 2-0: DMA2PR<2:0>: DMA2 Priority Selection bits

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

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-1/1</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SCANPR&lt;2:0&gt;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as ‘0’
- x = Bit is unknown
- -n/n = Value at POR and BOR/Value at all other Resets
- HS = Hardware set

**Legend:**
- bit 7-3: Unimplemented: Read as ‘0’
- bit 2-0: SCANPR<2:0>: Scanner Priority Selection bits

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

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PRLOCKED</td>
</tr>
</tbody>
</table>

**Legend:**
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as ‘0’
- x = Bit is unknown
- -n/n = Value at POR and BOR/Value at all other Resets
- HS = Hardware set

**Legend:**
- bit 7-1: Unimplemented: Read as ‘0’
- bit 0: PRLOCKED: PR Register Lock bit(1, 2)
  - 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 have access to the memory

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

**Note 2:** If PR1WAY = 1, the PRLOCKED bit cannot be cleared after it has been set. A device Reset will clear the bit and allow one more set.
TABLE 3-2: SUMMARY OF REGISTERS ASSOCIATED WITH CPU

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISRPR</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ISRPR2</td>
<td>ISRPR1</td>
<td>ISRPR0</td>
</tr>
<tr>
<td>MAINPR</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>MAINPR2</td>
<td>MAINPR1</td>
<td>MAINPR0</td>
</tr>
<tr>
<td>DMA1PR</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>DMA1PR2</td>
<td>DMA1PR1</td>
<td>DMA1PR0</td>
</tr>
<tr>
<td>DMA2PR</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>DMA2PR2</td>
<td>DMA2PR1</td>
<td>DMA2PR0</td>
</tr>
<tr>
<td>SCANPR</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>SCANPR2</td>
<td>SCANPR1</td>
<td>SCANPR0</td>
</tr>
<tr>
<td>PRLOCK</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>PRLOCKED</td>
</tr>
</tbody>
</table>

Legend: — = Unimplemented location, read as ‘0’.
4.0 MEMORY ORGANIZATION

There are three types of memory in PIC18 microcontroller devices:

• Program Flash Memory
• Data RAM
• Data EEPROM

As Harvard architecture devices, the data and program memories use separate buses; this allows for concurrent access of the two memory spaces. The data EEPROM, for practical purposes, can be regarded as a peripheral device, since it is addressed and accessed through a set of control registers.

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 contain the following:

• PIC18(L)F45/55K42: 32 Kbytes of Flash memory, up to 16,384 single-word instructions
• PIC18(L)F26/46/56K42: 64 Kbytes of Flash memory, up to 32,768 single-word instructions
• PIC18(L)F27/47/57K42: 128 Kbytes of Flash memory, up to 65,536 single-word instructions

The Reset vector for the device is at address 000000h. PIC18(L)F26/27/45/46/47/55/56/57K42 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 (BBEN = 1 and SAFEN = 1) assign all memory in the program Flash memory area to the application block. The 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 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 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 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 WRTSAF Configuration bit is used to write-protect the Storage Area Flash.

Note: If write-protected locations are written to, memory is not changed and the WRERR bit defined in Register 13-1 is set.
TABLE 4-1: PROGRAM AND DATA EEPROM MEMORY MAP

<table>
<thead>
<tr>
<th>PIC18(L)F45/55K42</th>
<th>PIC18(L)F26/46/56K42</th>
<th>PIC18(L)F27/47/57K42</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PC&lt;21:0&gt;</strong></td>
<td><strong>PC&lt;21:0&gt;</strong></td>
<td><strong>PC&lt;21:0&gt;</strong></td>
</tr>
<tr>
<td><strong>Note 1</strong></td>
<td><strong>Note 1</strong></td>
<td><strong>Note 1</strong></td>
</tr>
<tr>
<td>Stack (31 levels)</td>
<td>Stack (31 levels)</td>
<td>Stack (31 levels)</td>
</tr>
</tbody>
</table>

00 0000h

••• Reset Vector

••• Interrupt Vector High(2)

••• Interrupt Vector Low(2)

00 0008h

••• Interrupt Vector High(2)

••• Interrupt Vector Low(2)

00 0018h

••• Interrupt Vector Low(2)

00 001Ah

• 00 7FFFh

Program Flash Memory (16 KW)(3)

00 8000h

• 00 FFFFh

01 0000h

• 01 FFFFh

01 FFFFh

• 02 0000h

Reserved(4)

02 0000h

• 1F FFFFh

User IDs (8 Words)(5)

20 0000h

• 20 000Fh

20 0010h

• 2F FFFFh

30 0000h

• 30 0009h

30 000Ah

• 30 FFFFh

Data EEPROM (256 Bytes)

30 0000h

• 30 000Ah

31 0000h

• 31 00FFh

31 0100h

• 31 03FFh

31 0400h

• 3E FFFFh

3F 0000h

• 3F 003Fh

3F 0040h

• 3F FEFFh

3F FF00h

• 3F FF09h

3F FF0Ah

• 3F FFFBh

3F FFFCh

• 3F FFFDh

3F FFFEh

• 3F FFFFh

3F 0000h

• 3F 000Ah

3F 003Fh

• 3F 0040h

Device Information Area(5),(7)

3F 000Ah

• 3F 0040h

Device Configuration Information (5 Words)(5),(6),(7)

3F FF00h

• 3F FF09h

3F FFFBh

• 3F FFFFh

Revision ID (1 Word)(5),(6),(7)

3F FFFCh

• 3F FFFFh

Device ID (1 Word)(5),(6),(7)

3F FFFEh

• 3F FFFFh

**Note 1:** The stack is a separate SRAM panel, apart from all user memory panels.

**Note 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.

**Note 3:** Storage area Flash is implemented as the last 128 Words of user Flash.

**Note 4:** The addresses do not roll over. The region is read as ‘0’.

**Note 5:** Not code-protected.

**Note 6:** Hard-coded in silicon.

**Note 7:** This region cannot be written by the user and it’s not affected by a Bulk Erase.
### TABLE 4-2: PROGRAM FLASH MEMORY PARTITION

<table>
<thead>
<tr>
<th>Region</th>
<th>Address</th>
<th>Partition&lt;sup&gt;(3)&lt;/sup&gt;</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Program Flash Memory</td>
<td>00 0000h</td>
<td>BBEN = 1 SAFEN = 1</td>
<td>BBEN = 1 SAFEN = 0</td>
<td>BBEN = 0 SAFEN = 1</td>
<td>BBEN = 0 SAFEN = 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• • •</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Last Boot Block Memory Address</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Last Boot Block Memory Address&lt;sup&gt;(1)&lt;/sup&gt; + 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• • •</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Last Program Memory Address&lt;sup&gt;(2)&lt;/sup&gt; - 100h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Last Program Memory Address&lt;sup&gt;(2)&lt;/sup&gt; - FEh&lt;sup&gt;(4)&lt;/sup&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• • •</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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 4-1.

**3:** Refer to Register 5-7: Configuration Word 4L for BBEN and SAFEN definitions.

**4:** Storage area Flash is implemented as the last 128 Words of User Flash.
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.
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) 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.

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

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Legend</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>U-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>6</td>
<td>U-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>5</td>
<td>U-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>4</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>3</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>2</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>1</td>
<td>U-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>0</td>
<td>U-0</td>
<td>R/W-0</td>
</tr>
</tbody>
</table>

TOS<20:16>  

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

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Legend</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>6</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>5</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>4</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>3</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>2</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>1</td>
<td>U-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>0</td>
<td>U-0</td>
<td>R/W-0</td>
</tr>
</tbody>
</table>

TOS<15:8>  

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

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Legend</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>6</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>5</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>4</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>3</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>2</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>1</td>
<td>U-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>0</td>
<td>U-0</td>
<td>R/W-0</td>
</tr>
</tbody>
</table>

TOS<7:0>  

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
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

```
        MOVF 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 INSTRUCTION FLOW/PIPELINING
An “Instruction Cycle” consists of four cycles of the oscillator clock. 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).

EXAMPLE 4-3: INSTRUCTION PIPELINE FLOW

<table>
<thead>
<tr>
<th>TCY0</th>
<th>TCY1</th>
<th>TCY2</th>
<th>TCY3</th>
<th>TCY4</th>
<th>TCY5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fetch</td>
<td>Execute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. MOVW LW 55h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. MOVWF PORTB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. BRA SUB_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. BSF PORTA, BIT3 (Forced NOP)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. Instruction @ address SUB_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:** There are some instructions that take multiple cycles to execute. Refer to Section 41.0 “Instruction Set Summary” for details.
4.4.2 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 4-2 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 4-2 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 41.0 “Instruction Set Summary” provides further details of the instruction set.

4.4.3 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 MOVSFIL. 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 4-2: INSTRUCTIONS IN PROGRAM MEMORY](image-url)

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Byte Locations</th>
<th>Word Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instruction 1: MVVLW 055h</td>
<td>0Fh 55h 000008h</td>
<td>000000h</td>
</tr>
<tr>
<td>Instruction 2: GOTO 0006h</td>
<td>EFh 03h 000010h</td>
<td>000002h</td>
</tr>
<tr>
<td>Instruction 3: MOVFF 123h, 456h</td>
<td>C1h 23h 000012h</td>
<td>000004h</td>
</tr>
<tr>
<td>Instruction 4: MOVFFL 123h, 456h</td>
<td>F4h 56h 000014h</td>
<td>000006h</td>
</tr>
</tbody>
</table>

© 2017 Microchip Technology Inc. Preliminary DS40001919B-page 42
### EXAMPLE 4-4: TWO-WORD INSTRUCTIONS

**CASE 1:**

<table>
<thead>
<tr>
<th>Object Code</th>
<th>Source Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0110 0110 0000 0000</td>
<td>TSTFSZ REG1 ; is RAM location 0?</td>
</tr>
<tr>
<td>1100 0001 0010 0011</td>
<td>MOVFF REG1, REG2 ; Yes, skip this word</td>
</tr>
<tr>
<td>1111 0100 0101 0110</td>
<td>; Execute this word as a NOP</td>
</tr>
<tr>
<td>0010 0100 0000 0000</td>
<td>ADDWF REG3 ; continue code</td>
</tr>
</tbody>
</table>

**CASE 2:**

<table>
<thead>
<tr>
<th>Object Code</th>
<th>Source Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0110 0110 0000 0000</td>
<td>TSTFSZ REG1 ; is RAM location 0?</td>
</tr>
<tr>
<td>1100 0001 0010 0011</td>
<td>MOVFF REG1, REG2 ; No, execute this word</td>
</tr>
<tr>
<td>1111 0100 0101 0110</td>
<td>; 2nd word of instruction</td>
</tr>
<tr>
<td>0010 0100 0000 0000</td>
<td>ADDWF REG3 ; continue code</td>
</tr>
</tbody>
</table>

### EXAMPLE 4-5: THREE-WORD INSTRUCTIONS

**CASE 1:**

<table>
<thead>
<tr>
<th>Object Code</th>
<th>Source Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0110 0110 0000 0000</td>
<td>TSTFSZ REG1 ; is RAM location 0?</td>
</tr>
<tr>
<td>0000 0000 0110 0000</td>
<td>MOVFFL REG1, REG2 ; Yes, skip this word</td>
</tr>
<tr>
<td>1111 0100 1000 1100</td>
<td>; Execute this word as a NOP</td>
</tr>
<tr>
<td>1111 0100 0101 0110</td>
<td>; Execute this word as a NOP</td>
</tr>
<tr>
<td>0010 0100 0000 0000</td>
<td>ADDWF REG3 ; continue code</td>
</tr>
</tbody>
</table>

**CASE 2:**

<table>
<thead>
<tr>
<th>Object Code</th>
<th>Source Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0110 0110 0000 0000</td>
<td>TSTFSZ REG1 ; is RAM location 0?</td>
</tr>
<tr>
<td>0000 0000 0110 0000</td>
<td>MOVFFL REG1, REG2 ; No, execute this word</td>
</tr>
<tr>
<td>1111 0100 1000 1100</td>
<td>; 2nd word of instruction</td>
</tr>
<tr>
<td>1111 0100 0101 0110</td>
<td>; 3rd word of instruction</td>
</tr>
<tr>
<td>0010 0100 0000 0000</td>
<td>ADDWF REG3 ; continue code</td>
</tr>
</tbody>
</table>
4.5 Data Memory Organization

Data memory in PIC18F26/27/45/46/47/55/56/57K42 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 4-3 shows the data memory organization for the PIC18F26/27/45/46/47/55/56/57K42 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 4-3.

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 4-3 indicate which banks are implemented.
FIGURE 4-4: DATA MEMORY MAP FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES

<table>
<thead>
<tr>
<th>Bank</th>
<th>BSR&lt;5:0&gt;</th>
<th>Address addr&lt;7:0&gt;</th>
<th>PIC18(L)F45K42</th>
<th>PIC18(L)F26K42</th>
<th>PIC18(L)F46K42</th>
<th>PIC18(L)F56K42</th>
<th>PIC18(L)F57K42</th>
<th>Address addr&lt;13:0&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bank 0</td>
<td>00 0000</td>
<td>00h</td>
<td>Access RAM</td>
<td>Access RAM</td>
<td>Access RAM</td>
<td>0000h</td>
<td>005Fh</td>
<td></td>
</tr>
<tr>
<td></td>
<td>00 0001</td>
<td>00h</td>
<td>GPR</td>
<td>GPR</td>
<td>GPR</td>
<td>0060h</td>
<td>00FFh</td>
<td></td>
</tr>
<tr>
<td>Bank 2</td>
<td>00 0100</td>
<td>00h</td>
<td>GPR</td>
<td>GPR</td>
<td>GPR</td>
<td>0100h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bank 3</td>
<td>00 0011</td>
<td>00h</td>
<td>GPR</td>
<td>GPR</td>
<td>GPR</td>
<td>03FFh</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bank 4 to 7</td>
<td>00 0100</td>
<td>00h</td>
<td>GPR</td>
<td>GPR</td>
<td>GPR</td>
<td>0400h</td>
<td>07FFh</td>
<td></td>
</tr>
<tr>
<td>Bank 8 to 15</td>
<td>00 1000</td>
<td>00h</td>
<td>GPR</td>
<td>GPR</td>
<td>GPR</td>
<td>0800h</td>
<td>0FFFh</td>
<td></td>
</tr>
<tr>
<td>Bank 16 to 31</td>
<td>01 0000</td>
<td>00h</td>
<td>Unimplemented</td>
<td>Unimplemented</td>
<td>GPR</td>
<td>1000h</td>
<td>1FFFh</td>
<td></td>
</tr>
<tr>
<td>Bank 32 to 55</td>
<td>10 0000</td>
<td>00h</td>
<td>Unimplemented</td>
<td>Unimplemented</td>
<td>Unimplemented</td>
<td>2000h</td>
<td>37FFh</td>
<td></td>
</tr>
<tr>
<td>Bank 56 to 62</td>
<td>11 0000</td>
<td>00h</td>
<td>SFR</td>
<td>SFR</td>
<td>SFR</td>
<td>3800h</td>
<td>3EFFh</td>
<td></td>
</tr>
<tr>
<td>Bank 63</td>
<td>11 1111</td>
<td>00h</td>
<td>SFR</td>
<td>SFR</td>
<td>SFR</td>
<td>3800h</td>
<td>3FFFh</td>
<td></td>
</tr>
</tbody>
</table>
Note 1: The Access RAM bit of the instruction can be used to force an override of the selected bank (BSR<5:0>) to the registers of the Access Bank.
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 4-11. A bitwise summary of these registers can be found in Section 42.0 “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 4-4).

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”.

© 2017 Microchip Technology Inc. Preliminary DS40001919B-page 47
## TABLE 4-3: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES BANK 64

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>40FFh</td>
<td></td>
</tr>
<tr>
<td>40E9h</td>
<td></td>
</tr>
<tr>
<td>40E8h</td>
<td></td>
</tr>
<tr>
<td>40E7h</td>
<td></td>
</tr>
<tr>
<td>40E6h</td>
<td></td>
</tr>
<tr>
<td>40E5h</td>
<td></td>
</tr>
<tr>
<td>40E4h</td>
<td></td>
</tr>
<tr>
<td>40E3h</td>
<td></td>
</tr>
<tr>
<td>40E2h</td>
<td></td>
</tr>
<tr>
<td>40E1h</td>
<td></td>
</tr>
<tr>
<td>40E0h</td>
<td></td>
</tr>
<tr>
<td>40F0h</td>
<td></td>
</tr>
<tr>
<td>40F1h</td>
<td></td>
</tr>
<tr>
<td>40F2h</td>
<td></td>
</tr>
<tr>
<td>40F3h</td>
<td></td>
</tr>
<tr>
<td>40F4h</td>
<td></td>
</tr>
<tr>
<td>40F5h</td>
<td></td>
</tr>
<tr>
<td>40F6h</td>
<td></td>
</tr>
<tr>
<td>40F7h</td>
<td></td>
</tr>
<tr>
<td>40F8h</td>
<td></td>
</tr>
<tr>
<td>40F9h</td>
<td></td>
</tr>
<tr>
<td>40FAh</td>
<td></td>
</tr>
<tr>
<td>40FBh</td>
<td></td>
</tr>
<tr>
<td>40FCh</td>
<td></td>
</tr>
<tr>
<td>40FDh</td>
<td></td>
</tr>
<tr>
<td>40FEh</td>
<td></td>
</tr>
<tr>
<td>40FFh</td>
<td></td>
</tr>
</tbody>
</table>

Note 1: Addresses in Bank 64 are accessible ONLY through DMA Source and Destination Address Registers. CPU does not have access to registers in Bank 64.
<table>
<thead>
<tr>
<th>Address</th>
<th>Register Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3FFFh</td>
<td>TOSU</td>
<td>INDF2</td>
</tr>
<tr>
<td>3FFBh</td>
<td>PCLATU</td>
<td>PLUSW2</td>
</tr>
<tr>
<td>3FFEh</td>
<td>POSTINC0</td>
<td>PORTB</td>
</tr>
<tr>
<td>3F2A0h</td>
<td>FSRO0H</td>
<td>BSR</td>
</tr>
<tr>
<td>3F29h</td>
<td>FSRL0</td>
<td>TRISA</td>
</tr>
<tr>
<td>3F27h</td>
<td>WREG</td>
<td>TRISC</td>
</tr>
<tr>
<td>3FFC0h</td>
<td>POSTDEC0</td>
<td>TRISD</td>
</tr>
<tr>
<td>3FEC1h</td>
<td>POSTINC1</td>
<td>TRISB</td>
</tr>
<tr>
<td>3FF0h</td>
<td>PCON0</td>
<td>TRISC</td>
</tr>
<tr>
<td>3FF9h</td>
<td>PCON1</td>
<td>TRISB</td>
</tr>
<tr>
<td>3FF8h</td>
<td>POSTDEC1</td>
<td>TRISD</td>
</tr>
<tr>
<td>3FF7h</td>
<td>TBLPTRL</td>
<td>TRISB</td>
</tr>
<tr>
<td>3FF6h</td>
<td>TBLPRH</td>
<td>TRISD</td>
</tr>
<tr>
<td>3FF5h</td>
<td>TABLA</td>
<td>TRISD</td>
</tr>
<tr>
<td>3FF4h</td>
<td>PRODH</td>
<td>TRISD</td>
</tr>
<tr>
<td>3FF3h</td>
<td>PRODL</td>
<td>TRISD</td>
</tr>
<tr>
<td>3FF2h</td>
<td>—</td>
<td>TRISD</td>
</tr>
<tr>
<td>3FFFh</td>
<td>INDF1</td>
<td>TRISB</td>
</tr>
<tr>
<td>3FBFh</td>
<td>LATF</td>
<td>TRISD</td>
</tr>
<tr>
<td>3FF9h</td>
<td>T4PR</td>
<td>TRISD</td>
</tr>
<tr>
<td>3FF8h</td>
<td>CCP1CON</td>
<td>TRISD</td>
</tr>
<tr>
<td>3FF7h</td>
<td>CCP1TMRS1</td>
<td>TRISD</td>
</tr>
<tr>
<td>3FF6h</td>
<td>NCOTCLK</td>
<td>TRISD</td>
</tr>
<tr>
<td>3FFFh</td>
<td>SMT1CON1</td>
<td>TRISD</td>
</tr>
</tbody>
</table>

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

Note 1: Unimplemented in LF devices.
Note 2: Unimplemented in PIC18(L)F26/27K42.
<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Description</th>
<th>Address</th>
<th>Name</th>
<th>Description</th>
<th>Address</th>
<th>Name</th>
<th>Description</th>
<th>Address</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3EEFh</td>
<td>ADCLK</td>
<td></td>
<td>3EDFh</td>
<td>ADLTHH</td>
<td></td>
<td>3EBFh</td>
<td>CM1PCH</td>
<td></td>
<td>3E9Fh</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3EFDh</td>
<td>ADAC</td>
<td></td>
<td>3EDCH</td>
<td>ADLTHL</td>
<td></td>
<td>3EBEH</td>
<td>CM1NCH</td>
<td></td>
<td>3E9EH</td>
<td>DACKON</td>
<td>3E7EH</td>
</tr>
<tr>
<td>3ECFh</td>
<td>ADSTAT</td>
<td></td>
<td>3EDDH</td>
<td>ADCON3</td>
<td></td>
<td>3EBEH</td>
<td>CM2PCH</td>
<td></td>
<td>3E9BH</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3EF Bh</td>
<td>ADCON2</td>
<td></td>
<td>3EDC H</td>
<td>ADCON1</td>
<td></td>
<td>3EBAH</td>
<td>CM2NCH</td>
<td></td>
<td>3E9AH</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3EF Ah</td>
<td>ADCON0</td>
<td></td>
<td>3EDB H</td>
<td>ADCON0</td>
<td></td>
<td>3EBE H</td>
<td>CM2CON0</td>
<td></td>
<td>3E9EH</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E Fh</td>
<td>ADPREH</td>
<td></td>
<td>3ED A H</td>
<td>ADPREL</td>
<td></td>
<td>3EBH H</td>
<td>ADPCH</td>
<td></td>
<td>3E9BH</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E Fh</td>
<td>ADPREL</td>
<td></td>
<td>3ED9 H</td>
<td>ADCAP</td>
<td></td>
<td>3EBH H</td>
<td>ADPCH</td>
<td></td>
<td>3E99H</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E Fh</td>
<td>ADPREL</td>
<td></td>
<td>3ED8h</td>
<td>ADAC</td>
<td></td>
<td>3EBH H</td>
<td>ADPCH</td>
<td></td>
<td>3E98h</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E Fh</td>
<td>ADAC</td>
<td></td>
<td>3ED7h</td>
<td>ADPREH</td>
<td></td>
<td>3EBH H</td>
<td>ADPCH</td>
<td></td>
<td>3E97h</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E Fh</td>
<td>ADPREH</td>
<td></td>
<td>3ED6h</td>
<td>ADPREL</td>
<td></td>
<td>3EBH H</td>
<td>ADPCH</td>
<td></td>
<td>3E96h</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E Fh</td>
<td>ADPREL</td>
<td></td>
<td>3ED5h</td>
<td>ADAC</td>
<td></td>
<td>3EBH H</td>
<td>ADPCH</td>
<td></td>
<td>3E95h</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E Fh</td>
<td>ADAC</td>
<td></td>
<td>3ED4h</td>
<td>ADPREL</td>
<td></td>
<td>3EBH H</td>
<td>ADPCH</td>
<td></td>
<td>3E94h</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E Fh</td>
<td>ADPREL</td>
<td></td>
<td>3ED3h</td>
<td>ADAC</td>
<td></td>
<td>3EBH H</td>
<td>ADPCH</td>
<td></td>
<td>3E93h</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E Fh</td>
<td>ADAC</td>
<td></td>
<td>3ED2h</td>
<td>ADPREL</td>
<td></td>
<td>3EBH H</td>
<td>ADPCH</td>
<td></td>
<td>3E92h</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E Fh</td>
<td>ADAC</td>
<td></td>
<td>3ED1h</td>
<td>ADPREL</td>
<td></td>
<td>3EBH H</td>
<td>ADPCH</td>
<td></td>
<td>3E91h</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E Fh</td>
<td>ADAC</td>
<td></td>
<td>3ED0h</td>
<td>ADPREL</td>
<td></td>
<td>3EBH H</td>
<td>ADPCH</td>
<td></td>
<td>3E90h</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E FDh</td>
<td>ADRESH</td>
<td></td>
<td>3ED9h</td>
<td>ADRESL</td>
<td></td>
<td>3EBF H</td>
<td>ADSTPCH</td>
<td></td>
<td>3E9F H</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E FDh</td>
<td>ADRESL</td>
<td></td>
<td>3ED8h</td>
<td>ADRESL</td>
<td></td>
<td>3EBF H</td>
<td>ADSTPCH</td>
<td></td>
<td>3E98h</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E FDh</td>
<td>ADRESL</td>
<td></td>
<td>3ED7h</td>
<td>ADRESL</td>
<td></td>
<td>3EBF H</td>
<td>ADSTPCH</td>
<td></td>
<td>3E97h</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E FDh</td>
<td>ADRESL</td>
<td></td>
<td>3ED6h</td>
<td>ADRESL</td>
<td></td>
<td>3EBF H</td>
<td>ADSTPCH</td>
<td></td>
<td>3E96h</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E FDh</td>
<td>ADRESL</td>
<td></td>
<td>3ED5h</td>
<td>ADRESL</td>
<td></td>
<td>3EBF H</td>
<td>ADSTPCH</td>
<td></td>
<td>3E95h</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E FDh</td>
<td>ADRESL</td>
<td></td>
<td>3ED4h</td>
<td>ADRESL</td>
<td></td>
<td>3EBF H</td>
<td>ADSTPCH</td>
<td></td>
<td>3E94h</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E FDh</td>
<td>ADRESL</td>
<td></td>
<td>3ED3h</td>
<td>ADRESL</td>
<td></td>
<td>3EBF H</td>
<td>ADSTPCH</td>
<td></td>
<td>3E93h</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E FDh</td>
<td>ADRESL</td>
<td></td>
<td>3ED2h</td>
<td>ADRESL</td>
<td></td>
<td>3EBF H</td>
<td>ADSTPCH</td>
<td></td>
<td>3E92h</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E FDh</td>
<td>ADRESL</td>
<td></td>
<td>3ED1h</td>
<td>ADRESL</td>
<td></td>
<td>3EBF H</td>
<td>ADSTPCH</td>
<td></td>
<td>3E91h</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E FDh</td>
<td>ADRESL</td>
<td></td>
<td>3ED0h</td>
<td>ADRESL</td>
<td></td>
<td>3EBF H</td>
<td>ADSTPCH</td>
<td></td>
<td>3E90h</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E FCh</td>
<td>ADRESH</td>
<td></td>
<td>3ED9h</td>
<td>ADRESH</td>
<td></td>
<td>3EBE H</td>
<td>ADSTPTL</td>
<td></td>
<td>3E9F H</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E FCh</td>
<td>ADRESH</td>
<td></td>
<td>3ED8h</td>
<td>ADRESH</td>
<td></td>
<td>3EBE H</td>
<td>ADSTPTL</td>
<td></td>
<td>3E98h</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E FCh</td>
<td>ADRESH</td>
<td></td>
<td>3ED7h</td>
<td>ADRESH</td>
<td></td>
<td>3EBE H</td>
<td>ADSTPTL</td>
<td></td>
<td>3E97h</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E FCh</td>
<td>ADRESH</td>
<td></td>
<td>3ED6h</td>
<td>ADRESH</td>
<td></td>
<td>3EBE H</td>
<td>ADSTPTL</td>
<td></td>
<td>3E96h</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E FCh</td>
<td>ADRESH</td>
<td></td>
<td>3ED5h</td>
<td>ADRESH</td>
<td></td>
<td>3EBE H</td>
<td>ADSTPTL</td>
<td></td>
<td>3E95h</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E FCh</td>
<td>ADRESH</td>
<td></td>
<td>3ED4h</td>
<td>ADRESH</td>
<td></td>
<td>3EBE H</td>
<td>ADSTPTL</td>
<td></td>
<td>3E94h</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E FCh</td>
<td>ADRESH</td>
<td></td>
<td>3ED3h</td>
<td>ADRESH</td>
<td></td>
<td>3EBE H</td>
<td>ADSTPTL</td>
<td></td>
<td>3E93h</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E FCh</td>
<td>ADRESH</td>
<td></td>
<td>3ED2h</td>
<td>ADRESH</td>
<td></td>
<td>3EBE H</td>
<td>ADSTPTL</td>
<td></td>
<td>3E92h</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E FCh</td>
<td>ADRESH</td>
<td></td>
<td>3ED1h</td>
<td>ADRESH</td>
<td></td>
<td>3EBE H</td>
<td>ADSTPTL</td>
<td></td>
<td>3E91h</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E FCh</td>
<td>ADRESH</td>
<td></td>
<td>3ED0h</td>
<td>ADRESH</td>
<td></td>
<td>3EBE H</td>
<td>ADSTPTL</td>
<td></td>
<td>3E90h</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E FCh</td>
<td>ADRESH</td>
<td></td>
<td>3ED9h</td>
<td>ADRESH</td>
<td></td>
<td>3EBF H</td>
<td>ADUPTH</td>
<td></td>
<td>3E9F H</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E FCh</td>
<td>ADRESH</td>
<td></td>
<td>3ED8h</td>
<td>ADRESH</td>
<td></td>
<td>3EBF H</td>
<td>ADUPTH</td>
<td></td>
<td>3E98h</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E FCh</td>
<td>ADRESH</td>
<td></td>
<td>3ED7h</td>
<td>ADRESH</td>
<td></td>
<td>3EBF H</td>
<td>ADUPTH</td>
<td></td>
<td>3E97h</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E FCh</td>
<td>ADRESH</td>
<td></td>
<td>3ED6h</td>
<td>ADRESH</td>
<td></td>
<td>3EBF H</td>
<td>ADUPTH</td>
<td></td>
<td>3E96h</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E FCh</td>
<td>ADRESH</td>
<td></td>
<td>3ED5h</td>
<td>ADRESH</td>
<td></td>
<td>3EBF H</td>
<td>ADUPTH</td>
<td></td>
<td>3E95h</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E FCh</td>
<td>ADRESH</td>
<td></td>
<td>3ED4h</td>
<td>ADRESH</td>
<td></td>
<td>3EBF H</td>
<td>ADUPTH</td>
<td></td>
<td>3E94h</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E FCh</td>
<td>ADRESH</td>
<td></td>
<td>3ED3h</td>
<td>ADRESH</td>
<td></td>
<td>3EBF H</td>
<td>ADUPTH</td>
<td></td>
<td>3E93h</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E FCh</td>
<td>ADRESH</td>
<td></td>
<td>3ED2h</td>
<td>ADRESH</td>
<td></td>
<td>3EBF H</td>
<td>ADUPTH</td>
<td></td>
<td>3E92h</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E FCh</td>
<td>ADRESH</td>
<td></td>
<td>3ED1h</td>
<td>ADRESH</td>
<td></td>
<td>3EBF H</td>
<td>ADUPTH</td>
<td></td>
<td>3E91h</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3E FCh</td>
<td>ADRESH</td>
<td></td>
<td>3ED0h</td>
<td>ADRESH</td>
<td></td>
<td>3EBF H</td>
<td>ADUPTH</td>
<td></td>
<td>3E90h</td>
<td>___</td>
<td>___</td>
</tr>
</tbody>
</table>

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

Note 1: Unimplemented in LF devices.

Note 2: Unimplemented in PIC18(L)F26/27K42.

**TABLE 4-6: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES BANK 61**

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3DFFh</td>
<td>U2FIFO</td>
</tr>
<tr>
<td>3DFFh</td>
<td>3DBFh</td>
</tr>
<tr>
<td>3D9Fh</td>
<td>3D7Fh</td>
</tr>
<tr>
<td>3D5Fh</td>
<td>I2C2CON2</td>
</tr>
<tr>
<td>3D3Fh</td>
<td>3D1Fh</td>
</tr>
<tr>
<td>3D2Fh</td>
<td>I2C1CON2</td>
</tr>
<tr>
<td>3D1Fh</td>
<td>I2C2ADR3</td>
</tr>
<tr>
<td>3D0Fh</td>
<td>I2C2ADR0</td>
</tr>
<tr>
<td>3DFFh</td>
<td>U2BRGH</td>
</tr>
<tr>
<td>3DFFh</td>
<td>3DBEh</td>
</tr>
<tr>
<td>3D9Eh</td>
<td>3D7Eh</td>
</tr>
<tr>
<td>3D5Eh</td>
<td>I2C2ADR1</td>
</tr>
<tr>
<td>3D3Eh</td>
<td>I2C2ADR0</td>
</tr>
<tr>
<td>3D2Eh</td>
<td>I2C1BRG</td>
</tr>
<tr>
<td>3D1Eh</td>
<td>I2C1STAT0</td>
</tr>
<tr>
<td>3D0Eh</td>
<td>I2C1STAT1</td>
</tr>
<tr>
<td>3DFFh</td>
<td>U2BRGL</td>
</tr>
<tr>
<td>3DFFh</td>
<td>3DBDh</td>
</tr>
<tr>
<td>3D9Dh</td>
<td>3D7Dh</td>
</tr>
<tr>
<td>3D5Dh</td>
<td>I2C2TXB</td>
</tr>
<tr>
<td>3D3Dh</td>
<td>I2C2RXB</td>
</tr>
<tr>
<td>3D2Dh</td>
<td>I2C1P3H</td>
</tr>
<tr>
<td>3D1Dh</td>
<td>I2C1P3L</td>
</tr>
<tr>
<td>3D0Dh</td>
<td>I2C1RXB</td>
</tr>
<tr>
<td>3DFFh</td>
<td>U1TXCHK</td>
</tr>
<tr>
<td>3DFFh</td>
<td>3DCAh</td>
</tr>
<tr>
<td>3D9Ah</td>
<td>3D7Ah</td>
</tr>
<tr>
<td>3D5Ah</td>
<td>I2C1RXCB</td>
</tr>
<tr>
<td>3D3Ah</td>
<td>I2C1TXB</td>
</tr>
<tr>
<td>3D2Ah</td>
<td>U1RXB</td>
</tr>
<tr>
<td>3D1Ah</td>
<td>U1TXB</td>
</tr>
<tr>
<td>3D0Ah</td>
<td>U1P3H</td>
</tr>
<tr>
<td>3DFFh</td>
<td>U1PT3L</td>
</tr>
<tr>
<td>3DFFh</td>
<td>3DACf</td>
</tr>
<tr>
<td>3D9Fh</td>
<td>3D7Fh</td>
</tr>
<tr>
<td>3D5Fh</td>
<td>I2C1AD0</td>
</tr>
<tr>
<td>3D3Fh</td>
<td>I2C1AD4</td>
</tr>
<tr>
<td>3D2Fh</td>
<td>I2C1AD8</td>
</tr>
<tr>
<td>3D1Fh</td>
<td>I2C1AD12</td>
</tr>
<tr>
<td>3D0Fh</td>
<td>I2C1AD16</td>
</tr>
<tr>
<td>3DFFh</td>
<td>U1P1H</td>
</tr>
<tr>
<td>3DFFh</td>
<td>3DCAh</td>
</tr>
<tr>
<td>3D9Fh</td>
<td>3D7Fh</td>
</tr>
<tr>
<td>3D5Fh</td>
<td>I2C1AD0</td>
</tr>
<tr>
<td>3D3Fh</td>
<td>I2C1AD4</td>
</tr>
<tr>
<td>3D2Fh</td>
<td>I2C1AD8</td>
</tr>
<tr>
<td>3D1Fh</td>
<td>I2C1AD12</td>
</tr>
<tr>
<td>3D0Fh</td>
<td>I2C1AD16</td>
</tr>
<tr>
<td>3DFFh</td>
<td>U1P2H</td>
</tr>
<tr>
<td>3DFFh</td>
<td>3DCAh</td>
</tr>
<tr>
<td>3D9Fh</td>
<td>3D7Fh</td>
</tr>
<tr>
<td>3D5Fh</td>
<td>I2C1AD0</td>
</tr>
<tr>
<td>3D3Fh</td>
<td>I2C1AD4</td>
</tr>
<tr>
<td>3D2Fh</td>
<td>I2C1AD8</td>
</tr>
<tr>
<td>3D1Fh</td>
<td>I2C1AD12</td>
</tr>
<tr>
<td>3D0Fh</td>
<td>I2C1AD16</td>
</tr>
<tr>
<td>3DFFh</td>
<td>U1P1L</td>
</tr>
<tr>
<td>3DFFh</td>
<td>3DCAh</td>
</tr>
<tr>
<td>3D9Fh</td>
<td>3D7Fh</td>
</tr>
<tr>
<td>3D5Fh</td>
<td>I2C1AD0</td>
</tr>
<tr>
<td>3D3Fh</td>
<td>I2C1AD4</td>
</tr>
<tr>
<td>3D2Fh</td>
<td>I2C1AD8</td>
</tr>
<tr>
<td>3D1Fh</td>
<td>I2C1AD12</td>
</tr>
<tr>
<td>3D0Fh</td>
<td>I2C1AD16</td>
</tr>
<tr>
<td>3DFFh</td>
<td>U1TXCK</td>
</tr>
<tr>
<td>3DFFh</td>
<td>3DCAh</td>
</tr>
<tr>
<td>3D9Fh</td>
<td>3D7Fh</td>
</tr>
<tr>
<td>3D5Fh</td>
<td>I2C1TXB</td>
</tr>
<tr>
<td>3D3Fh</td>
<td>I2C1RXB</td>
</tr>
<tr>
<td>3D2Fh</td>
<td>U1P3L</td>
</tr>
<tr>
<td>3D1Fh</td>
<td>U1RXCB</td>
</tr>
<tr>
<td>3D0Fh</td>
<td>U1TXB</td>
</tr>
</tbody>
</table>

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

**Note:**
1. Unimplemented in LF devices.
2. Unimplemented in PIC18(L)F26/27K42.
**TABLE 4-7: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES BANK 60**

<table>
<thead>
<tr>
<th>Address</th>
<th>Register 1</th>
<th>Register 2</th>
<th>Register 3</th>
<th>Register 4</th>
<th>Register 5</th>
<th>Register 6</th>
<th>Register 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>3CFFh</td>
<td>3CDFh</td>
<td>3CBFh</td>
<td>3C9Fh</td>
<td>3C7Fh</td>
<td>3C5Fh</td>
<td>CLC4GLS3</td>
<td>3C3Fh</td>
</tr>
<tr>
<td>3CFEh</td>
<td>MDTCARH</td>
<td>3CDBh</td>
<td>3C8Fh</td>
<td>3C7Eh</td>
<td>CLCDATA0</td>
<td>CLC4GLS2</td>
<td>3C5Eh</td>
</tr>
<tr>
<td>3CFDh</td>
<td>MDTCARL</td>
<td>3CBCh</td>
<td>3C9Fh</td>
<td>3C7Dh</td>
<td>CLCGLS3</td>
<td>CLC4GLST</td>
<td>3C3Dh</td>
</tr>
<tr>
<td>3CFCh</td>
<td>MD1SRC</td>
<td>3CBnh</td>
<td>3C9Fh</td>
<td>3C7Ch</td>
<td>CLC1GLS2</td>
<td>CLC4GLS0</td>
<td>3C3Ch</td>
</tr>
<tr>
<td>3CFBh</td>
<td>MD1CON1</td>
<td>3CBBh</td>
<td>3C9Bh</td>
<td>3C7Bh</td>
<td>CLC1GLS1</td>
<td>CLC4SEL3</td>
<td>3C3Bh</td>
</tr>
<tr>
<td>3CFAh</td>
<td>MD1CON0</td>
<td>3CBAh</td>
<td>3C9Ah</td>
<td>3C7Ah</td>
<td>CLC1GLS0</td>
<td>CLC4SEL2</td>
<td>3C3Ah</td>
</tr>
<tr>
<td>3CF9h</td>
<td>3C8Fh</td>
<td>3C7Fh</td>
<td>3C3Fh</td>
<td>3C1Fh</td>
<td>3C1Fh</td>
<td>3C1Fh</td>
<td>3C1Fh</td>
</tr>
<tr>
<td>3CF8h</td>
<td>3C7Fh</td>
<td>3C6Fh</td>
<td>3C5Fh</td>
<td>3C4Fh</td>
<td>3C4Fh</td>
<td>3C4Fh</td>
<td>3C4Fh</td>
</tr>
<tr>
<td>3CF7h</td>
<td>3C6Fh</td>
<td>3C5Fh</td>
<td>3C4Fh</td>
<td>3C3Fh</td>
<td>3C3Fh</td>
<td>3C3Fh</td>
<td>3C3Fh</td>
</tr>
<tr>
<td>3CF6h</td>
<td>3C5Fh</td>
<td>3C4Fh</td>
<td>3C3Fh</td>
<td>3C2Fh</td>
<td>3C2Fh</td>
<td>3C2Fh</td>
<td>3C2Fh</td>
</tr>
<tr>
<td>3CF5h</td>
<td>3C4Fh</td>
<td>3C3Fh</td>
<td>3C2Fh</td>
<td>3C1Fh</td>
<td>3C1Fh</td>
<td>3C1Fh</td>
<td>3C1Fh</td>
</tr>
<tr>
<td>3CF4h</td>
<td>3C3Fh</td>
<td>3C2Fh</td>
<td>3C1Fh</td>
<td>3C0Fh</td>
<td>3C0Fh</td>
<td>3C0Fh</td>
<td>3C0Fh</td>
</tr>
<tr>
<td>3CF3h</td>
<td>3C2Fh</td>
<td>3C1Fh</td>
<td>3C0Fh</td>
<td>3CF9h</td>
<td>3CF9h</td>
<td>3CF9h</td>
<td>3CF9h</td>
</tr>
<tr>
<td>3CF2h</td>
<td>3C1Fh</td>
<td>3C0Fh</td>
<td>3CF9h</td>
<td>3CFFh</td>
<td>3CFFh</td>
<td>3CFFh</td>
<td>3CFFh</td>
</tr>
<tr>
<td>3CF1h</td>
<td>3C0Fh</td>
<td>3CF9h</td>
<td>3CFFh</td>
<td>3CFFh</td>
<td>3CFFh</td>
<td>3CFFh</td>
<td>3CFFh</td>
</tr>
<tr>
<td>3CF0h</td>
<td>3CF9h</td>
<td>3CFFh</td>
<td>3CFFh</td>
<td>3CFFh</td>
<td>3CFFh</td>
<td>3CFFh</td>
<td>3CFFh</td>
</tr>
<tr>
<td>3CFFh</td>
<td>3CFFh</td>
<td>3CFFh</td>
<td>3CFFh</td>
<td>3CFFh</td>
<td>3CFFh</td>
<td>3CFFh</td>
<td>3CFFh</td>
</tr>
</tbody>
</table>

**Legend:**
- Unimplemented data memory locations and registers, read as ‘0’.
- 1: Unimplemented in LF devices.
- 2: Unimplemented in PIC18(L)F26/27K42.
### TABLE 4-8: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES BANK 59

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
<th>Address</th>
<th>Description</th>
<th>Address</th>
<th>Description</th>
<th>Address</th>
<th>Description</th>
<th>Address</th>
<th>Description</th>
<th>Address</th>
<th>Description</th>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3BFH</td>
<td>DMA1SIRQ</td>
<td>3BFFH</td>
<td>DMA2SIRQ</td>
<td>3B9FH</td>
<td>DMA1SIRQ</td>
<td>3B5FH</td>
<td>DMA2SIRQ</td>
<td>3B3FH</td>
<td>DMA1SIRQ</td>
<td>3B1FH</td>
<td>DMA2SIRQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3BFEH</td>
<td>DMA1AIRQ</td>
<td>3BDFH</td>
<td>DMA2AIRQ</td>
<td>3BBFH</td>
<td>DMA1AIRQ</td>
<td>3B7FH</td>
<td>DMA2AIRQ</td>
<td>3B5BH</td>
<td>DMA1AIRQ</td>
<td>3B3BH</td>
<td>DMA2AIRQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3BFNH</td>
<td>DMA1CON1</td>
<td>3BCHF</td>
<td>DMA2CON1</td>
<td>3B9CH</td>
<td>DMA1CON1</td>
<td>3B5CH</td>
<td>DMA2CON1</td>
<td>3B3CH</td>
<td>DMA1CON1</td>
<td>3B1CH</td>
<td>DMA2CON1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3BF8H</td>
<td>DMA1SSAU</td>
<td>3BDFH</td>
<td>DMA2SSAU</td>
<td>3B9FH</td>
<td>DMA1SSAU</td>
<td>3B5FH</td>
<td>DMA2SSAU</td>
<td>3B3FH</td>
<td>DMA1SSAU</td>
<td>3B1FH</td>
<td>DMA2SSAU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3BFFH</td>
<td>DMA1SPIR</td>
<td>3BCHF</td>
<td>DMA2SPIR</td>
<td>3B9CH</td>
<td>DMA1SPIR</td>
<td>3B5CH</td>
<td>DMA2SPIR</td>
<td>3B3CH</td>
<td>DMA1SPIR</td>
<td>3B1CH</td>
<td>DMA2SPIR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3BEFH</td>
<td>DMA1SAPR</td>
<td>3BCHF</td>
<td>DMA2SAPR</td>
<td>3B9CH</td>
<td>DMA1SAPR</td>
<td>3B5CH</td>
<td>DMA2SAPR</td>
<td>3B3CH</td>
<td>DMA1SAPR</td>
<td>3B1CH</td>
<td>DMA2SAPR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3BE8H</td>
<td>DMA1SAPR</td>
<td>3BCHF</td>
<td>DMA2SAPR</td>
<td>3B9CH</td>
<td>DMA1SAPR</td>
<td>3B5CH</td>
<td>DMA2SAPR</td>
<td>3B3CH</td>
<td>DMA1SAPR</td>
<td>3B1CH</td>
<td>DMA2SAPR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3BEEH</td>
<td>DMA1SAPR</td>
<td>3BCHF</td>
<td>DMA2SAPR</td>
<td>3B9CH</td>
<td>DMA1SAPR</td>
<td>3B5CH</td>
<td>DMA2SAPR</td>
<td>3B3CH</td>
<td>DMA1SAPR</td>
<td>3B1CH</td>
<td>DMA2SAPR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3BEFH</td>
<td>DMA1SAPR</td>
<td>3BCHF</td>
<td>DMA2SAPR</td>
<td>3B9CH</td>
<td>DMA1SAPR</td>
<td>3B5CH</td>
<td>DMA2SAPR</td>
<td>3B3CH</td>
<td>DMA1SAPR</td>
<td>3B1CH</td>
<td>DMA2SAPR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3BEEH</td>
<td>DMA1SAPR</td>
<td>3BCHF</td>
<td>DMA2SAPR</td>
<td>3B9CH</td>
<td>DMA1SAPR</td>
<td>3B5CH</td>
<td>DMA2SAPR</td>
<td>3B3CH</td>
<td>DMA1SAPR</td>
<td>3B1CH</td>
<td>DMA2SAPR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3BEFH</td>
<td>DMA1SAPR</td>
<td>3BCHF</td>
<td>DMA2SAPR</td>
<td>3B9CH</td>
<td>DMA1SAPR</td>
<td>3B5CH</td>
<td>DMA2SAPR</td>
<td>3B3CH</td>
<td>DMA1SAPR</td>
<td>3B1CH</td>
<td>DMA2SAPR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3BEEH</td>
<td>DMA1SAPR</td>
<td>3BCHF</td>
<td>DMA2SAPR</td>
<td>3B9CH</td>
<td>DMA1SAPR</td>
<td>3B5CH</td>
<td>DMA2SAPR</td>
<td>3B3CH</td>
<td>DMA1SAPR</td>
<td>3B1CH</td>
<td>DMA2SAPR</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

**Note:**
1. Unimplemented in LF devices.
2. Unimplemented in PIC18(L)F26/27K42.
| 3AFFh — | 3ADFh | SP11SDDPPS | 3ABFh | PPSLOCK | 3AF9h — | 3A9Fh — | 3A5Fh — | 3A3Fh — | 3A1Fh | RD7PPS(2) |
| 3AF8h — | 3A9Fh | SP115CCPPS | 3AB8h — | 3A8Fh — | 3A5Eh — | 3A3Eh — | 3A1Eh | RD6PPS(2) |
| 3AFFh — | 3A9Fh | ADA1CTPPPS | 3AB7h — | 3A7Fh — | 3A5Dh — | 3A3Dh — | 3A1Dh | RD5PPS(2) |
| 3AFCN — | 3AACN | CLC1NPPPS | 3ABCN — | 3A9CN — | 3A5CN — | 3A3CN — | 3A1CN | RD4PPS(2) |
| 3AFBN — | 3A9BN | CLC1NPPPS | 3ABBN — | 3A8BN — | 3A7BN — | 3A5BN — | 3A3BN — | 3A1BN | RD3PPS(2) |
| 3AFLh — | 3A9FL | CLC1NPPPS | 3ABFL — | 3A8FL — | 3A7FL — | 3A5FL — | 3A3FL — | 3A1FL | RD2PPS(2) |
| 3AFNL — | 3A9NL | CLC1NPPPS | 3ABNL — | 3A8NL — | 3A7NL — | 3A5NL — | 3A3NL — | 3A1NL | RD1PPS(2) |
| 3AFLF — | 3A9LF | CLC1NPPPS | 3ABLF — | 3A8LF — | 3A7LF — | 3A5LF — | 3A3LF — | 3A1LF | R0PPS(2) |
| 3AFAF — | 3A9AF | CLC1NPPPS | 3ABAF — | 3A8AF — | 3A7AF — | 3A5AF — | 3A3AF — | 3A1AF | R0PPS(2) |
| 3AFF — | 3A9F | CLC1NPPPS | 3ABF — | 3A8F — | 3A7F — | 3A5F — | 3A3F — | 3A1F | R0PPS(2) |
| 3AF — | 3A9 | CLC1NPPPS | 3AB — | 3A8 — | 3A7 — | 3A5 — | 3A3 — | 3A1 | R0PPS(2) |
| 3A — | 3A | CLC1NPPPS | 3A — | 3A — | 3A — | 3A — | 3A — | 3A | R0PPS(2) |

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

Note 1: Unimplemented in LF devices.
2: Unimplemented in PIC18(L)F26/27K42.
4: Reserved, maintain as '0'.
### TABLE 4-10: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F26/27/45/46/55/56/57K42 DEVICES BANK 57

<table>
<thead>
<tr>
<th>Address</th>
<th>Register Name</th>
<th>Register Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>39FFh</td>
<td>OSCFRQ</td>
<td>39BFh</td>
</tr>
<tr>
<td>39F0h</td>
<td>OSCFRQ</td>
<td>39AFh</td>
</tr>
<tr>
<td>39F1h</td>
<td>OSCFRQ</td>
<td>399Fh</td>
</tr>
<tr>
<td>39F2h</td>
<td>OSCFRQ</td>
<td>398Fh</td>
</tr>
<tr>
<td>39F3h</td>
<td>OSCFRQ</td>
<td>397Fh</td>
</tr>
<tr>
<td>39F4h</td>
<td>OSCFRQ</td>
<td>396Fh</td>
</tr>
<tr>
<td>39F5h</td>
<td>OSCFRQ</td>
<td>395Fh</td>
</tr>
<tr>
<td>39F6h</td>
<td>OSCFRQ</td>
<td>394Fh</td>
</tr>
<tr>
<td>39F7h</td>
<td>OSCFRQ</td>
<td>393Fh</td>
</tr>
<tr>
<td>39F8h</td>
<td>OSCFRQ</td>
<td>392Fh</td>
</tr>
<tr>
<td>39F9h</td>
<td>OSCFRQ</td>
<td>391Fh</td>
</tr>
<tr>
<td>39FAh</td>
<td>OSCFRQ</td>
<td>390Fh</td>
</tr>
<tr>
<td>39FBh</td>
<td>OSCFRQ</td>
<td>3999h</td>
</tr>
<tr>
<td>39FCh</td>
<td>OSCFRQ</td>
<td>3989h</td>
</tr>
<tr>
<td>39FDh</td>
<td>OSCFRQ</td>
<td>3979h</td>
</tr>
<tr>
<td>39FEh</td>
<td>OSCFRQ</td>
<td>3969h</td>
</tr>
<tr>
<td>39FFh</td>
<td>OSCFRQ</td>
<td>3959h</td>
</tr>
<tr>
<td>3900h</td>
<td>OSCFRQ</td>
<td>3949h</td>
</tr>
<tr>
<td>3901h</td>
<td>OSCFRQ</td>
<td>3939h</td>
</tr>
<tr>
<td>3902h</td>
<td>OSCFRQ</td>
<td>3929h</td>
</tr>
<tr>
<td>3903h</td>
<td>OSCFRQ</td>
<td>3919h</td>
</tr>
<tr>
<td>3904h</td>
<td>OSCFRQ</td>
<td>3909h</td>
</tr>
</tbody>
</table>

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

Note 1: Unimplemented in LF devices.
Note 2: Unimplemented in PIC18(L)F26/27K42.
Note 4: Unimplemented in PIC18(L)F45/55K42.
### TABLE 4-11: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES BANK 56

<table>
<thead>
<tr>
<th>Register</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>38FFh</td>
<td>38DFFh</td>
</tr>
<tr>
<td>38F0h</td>
<td>38D0h</td>
</tr>
<tr>
<td>38F1h</td>
<td>38D1h</td>
</tr>
<tr>
<td>38F2h</td>
<td>38D2h</td>
</tr>
<tr>
<td>38F3h</td>
<td>38D3h</td>
</tr>
<tr>
<td>38F4h</td>
<td>38D4h</td>
</tr>
<tr>
<td>38F5h</td>
<td>38D5h</td>
</tr>
<tr>
<td>38F6h</td>
<td>38D6h</td>
</tr>
<tr>
<td>38F7h</td>
<td>38D7h</td>
</tr>
<tr>
<td>38F8h</td>
<td>38D8h</td>
</tr>
<tr>
<td>38F9h</td>
<td>38D9h</td>
</tr>
<tr>
<td>38FAh</td>
<td>38DAh</td>
</tr>
<tr>
<td>38FBh</td>
<td>38DBh</td>
</tr>
<tr>
<td>38FC</td>
<td>38DC</td>
</tr>
<tr>
<td>38FDh</td>
<td>38DDh</td>
</tr>
<tr>
<td>38FEh</td>
<td>38DEh</td>
</tr>
<tr>
<td>38FFh</td>
<td>38DFh</td>
</tr>
</tbody>
</table>

Legend: Unimplemented data memory locations and registers, read as ‘0’.

Note 1: Unimplemented in LF devices.
Note 2: Unimplemented in PIC18(L)F26/27K42.
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 uuuu').

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 41.2 “Extended Instruction Set” and Table 41-3.

**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. The contents of these registers should be handled correctly to avoid erroneous code execution.
### 4.6 Register Definitions: Status Registers

**REGISTER 4-2: STATUS: STATUS REGISTER**

<table>
<thead>
<tr>
<th>U-0</th>
<th>R-1/q</th>
<th>R-1/q</th>
<th>R/W-0/u</th>
<th>R/W-0/u</th>
<th>R/W-0/u</th>
<th>R/W-0/u</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TØ</td>
<td>PD</td>
<td>N</td>
<td>OV</td>
<td>Z</td>
<td>DC</td>
</tr>
</tbody>
</table>

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’
  - **TØ:** Time-Out bit
    - 1 = Set at power-up or by execution of **CLR WDT** or **SLEEP** instruction
    - 0 = A WDT time-out occurred

- **bit 6**
  - **PD:** Power-Down bit
    - 1 = Set at power-up or by execution of **CLR WDT** instruction
    - 0 = Set by execution of the **SLEEP** instruction

- **bit 5**
  - **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 4**
  - **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 3**
  - **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 2**
  - **DC:** Digit Carry/Borrow bit (**ADDWF**, **ADDLW**, **SUBLW**, **SUBWF** instructions)
    - 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 1**
  - **C:** Carry/Borrow bit (**ADDWF**, **ADDLW**, **SUBLW**, **SUBWF** instructions)
    - 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, 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.

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.
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**

Using an instruction with one of the indirect addressing registers as the operand....

...uses the 14-bit address stored in the FSR pair associated with that register....

...to determine the data memory location to be used in that operation.

In this case, the FSR1 pair contains 3ECCh. This means the contents of location 3ECCh will be added to that of the W register and stored back in 3ECCh.
Operations on the FSRs with POSTDEC, POSTINC and PREINC affect the entire register pair; that is, rollovers 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 41.2.1 “Extended Instruction Syntax”.

© 2017 Microchip Technology Inc.
EXAMPLE INSTRUCTION: ADDWF, f, d, a (Opcode: 0010 01da ffff ffff)

When 'a' = 0 and f ≥ 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 ≤ 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 41.2 “Extended Instruction Set”.

---

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

Example Situation:

ADDWF f, d, a
FSR2H:FSR2L = 120h

Locations in the region from the FSR2 pointer (0120h) to the pointer plus 05Fh (017Fh) are mapped to the bottom of the Access RAM (000h-05Fh).

Special File Registers at 3F60h through 3FFFh are mapped to 60h through FFh, as usual.

Bank 0 addresses below 5Fh can still be addressed by using the BSR.
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 300000h.
5.2 Register Definitions: Configuration Words

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

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 6-4</th>
<th>bit 3</th>
<th>bit 2-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>RSTOSC&lt;2:0&gt;</td>
<td>—</td>
<td>FEXTOSC&lt;2:0&gt;</td>
</tr>
</tbody>
</table>

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)(1)
- 110 = ECM (External Clock Medium Power)(1)
- 101 = ECL (External Clock Low Power)(1)
- 100 = Oscillator is not enabled
- 011 = Reserved (do not use)
- 010 = HS (crystal oscillator) above 8 MHz
- 001 = XT (crystal oscillator) above 500 kHz, below 8 MHz
- 000 = LP (crystal oscillator) optimized for 32.768 kHz

Note 1: Refer to Table 44-9 for External Clock/Oscillator Timing Requirements.
## REGISTER 5-2: CONFIGURATION WORD 1H (30 0001h)

<table>
<thead>
<tr>
<th>bit 7-6</th>
<th>bit 5</th>
<th>bit 4</th>
<th>bit 3</th>
<th>bit 2</th>
<th>bit 1</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unimplemented: Read as ‘1’</td>
<td>FCMEN: Fail-Safe Clock Monitor Enable bit</td>
<td>CSWEN: Clock Switch Enable bit</td>
<td>Unimplemented: Read as ‘1’</td>
<td>Unimplemented: Read as ‘1’</td>
<td>PR1WAY: PRLOCKED One-Way Set Enable bit</td>
<td>CLKOUTEN: Clock Out Enable bit</td>
</tr>
<tr>
<td></td>
<td>1 = FSCM timer is enabled</td>
<td>1 = Writing to NOSC and NDIV is allowed</td>
<td></td>
<td></td>
<td>1 = PRLOCKED bit can be cleared and set only once; Priority registers remain locked after one clear/set cycle</td>
<td>If FEXTOSC&lt;2:0&gt; = EC (high, mid or low) or Not Enabled:</td>
</tr>
<tr>
<td></td>
<td>0 = FSCM timer is disabled</td>
<td>0 = The NOSC and NDIV bits cannot be changed by user software</td>
<td></td>
<td></td>
<td>0 = PRLOCKED bit can be set and cleared multiple times (subject to the unlock sequence)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Otherwise:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This bit is ignored.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Bit 7-6</th>
<th>BOREN&lt;1:0&gt;: Brown-out Reset Enable bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7-0</td>
<td>R/W-1</td>
</tr>
<tr>
<td>bit 7</td>
<td>BOREN&lt;1:0&gt;</td>
</tr>
<tr>
<td>bit 6</td>
<td>LPBREN</td>
</tr>
<tr>
<td>bit 5</td>
<td>IVT1WAY</td>
</tr>
<tr>
<td>bit 4</td>
<td>MVECEN</td>
</tr>
<tr>
<td>bit 2-1</td>
<td>PWRTS&lt;1:0&gt;</td>
</tr>
<tr>
<td>bit 0</td>
<td>MCLRE</td>
</tr>
</tbody>
</table>

#### 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  LPBREN:** 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 = IVTLOCKED 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 (2048 LFINTOSC Cycles)
- 01 = PWRT set at 16 ms (512 LFINTOSC Cycles)
- 00 = PWRT set at 1 ms (32 LFINTOSC Cycles)

**bit 0  MCLRE:** Master Clear (MCLR) Enable bit
- If LVP = 1:
  - RE3 pin function is MCLR
- If LVP = 0:
  - 1 = MCLR pin is MCLR
  - 0 = MCLR pin function is a port defined function
### REGISTER 5-4: CONFIGURATION WORD 2H (30 0003h)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>XINST: Extended Instruction Set Enable bit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = Extended instruction set and Indexed Addressing mode are disabled (Legacy mode)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = Extended instruction set and Indexed Addressing mode are enabled</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Unimplemented: Read as ‘1’</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>DEBUG: Debugger Enable bit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = Background debugger is disabled</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = Background debugger is enabled</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>STVREN: Stack Overflow/Underflow Reset Enable bit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = Stack Overflow or Underflow will cause a Reset</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = Stack Overflow or Underflow will not cause a Reset</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>PPS1WAY: PPSLOCKED One-Way Set Enable bit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = PPSLOCKED bit can be cleared and set only once; PPS registers remain locked after one clear/set cycle</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = PPSLOCKED bit can be set and cleared multiple times (subject to the unlock sequence)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>ZCD: Zero-Cross Detect Enable bit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = ZCD is disabled; ZCD can be enabled by setting the bit SEN of the ZCDCON register</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = ZCD is always enabled</td>
<td></td>
</tr>
<tr>
<td>1-0</td>
<td>BORV&lt;1:0&gt;: Brown-out Reset Voltage Selection bits(1)</td>
<td></td>
</tr>
</tbody>
</table>

**Note 1:** The higher voltage setting is recommended for operation at or above 16 MHz.
### REGISTER 5-5: CONFIGURATION WORD 3L (30 0004h)

<table>
<thead>
<tr>
<th>U-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>WDTE&lt;1:0&gt;</td>
</tr>
</tbody>
</table>

#### 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

<table>
<thead>
<tr>
<th>WDTCPS&lt;4:0&gt;</th>
<th>WDTPS at POR</th>
<th>Typical Time-out (FIN = 31 kHz)</th>
<th>Software Control of WDTPS?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>Divider Ratio</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000</td>
<td>00000</td>
<td>1:32</td>
<td>2(^5)</td>
</tr>
<tr>
<td>00001</td>
<td>00001</td>
<td>1:64</td>
<td>2(^6)</td>
</tr>
<tr>
<td>00010</td>
<td>00010</td>
<td>1:128</td>
<td>2(^7)</td>
</tr>
<tr>
<td>00011</td>
<td>00011</td>
<td>1:256</td>
<td>2(^8)</td>
</tr>
<tr>
<td>00100</td>
<td>00100</td>
<td>1:512</td>
<td>2(^9)</td>
</tr>
<tr>
<td>00101</td>
<td>00101</td>
<td>1:1024</td>
<td>2(^{10})</td>
</tr>
<tr>
<td>00110</td>
<td>00110</td>
<td>1:2048</td>
<td>2(^{11})</td>
</tr>
<tr>
<td>00111</td>
<td>00111</td>
<td>1:4096</td>
<td>2(^{12})</td>
</tr>
<tr>
<td>01000</td>
<td>01000</td>
<td>1:8192</td>
<td>2(^{13})</td>
</tr>
<tr>
<td>01001</td>
<td>01001</td>
<td>1:16384</td>
<td>2(^{14})</td>
</tr>
<tr>
<td>01010</td>
<td>01010</td>
<td>1:32768</td>
<td>2(^{15})</td>
</tr>
<tr>
<td>01011</td>
<td>01011</td>
<td>1:65536</td>
<td>2(^{16})</td>
</tr>
<tr>
<td>01100</td>
<td>01100</td>
<td>1:131072</td>
<td>2(^{17})</td>
</tr>
<tr>
<td>01101</td>
<td>01101</td>
<td>1:262144</td>
<td>2(^{18})</td>
</tr>
<tr>
<td>01110</td>
<td>01110</td>
<td>1:524299</td>
<td>2(^{19})</td>
</tr>
<tr>
<td>01111</td>
<td>01111</td>
<td>1:1048576</td>
<td>2(^{20})</td>
</tr>
<tr>
<td>10000</td>
<td>10000</td>
<td>1:2097152</td>
<td>2(^{21})</td>
</tr>
<tr>
<td>10001</td>
<td>10001</td>
<td>1:4194304</td>
<td>2(^{22})</td>
</tr>
<tr>
<td>10010</td>
<td>10010</td>
<td>1:8388608</td>
<td>2(^{23})</td>
</tr>
<tr>
<td>10011</td>
<td>10011</td>
<td>1:65536</td>
<td>2(^{16})</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>1:32</td>
<td>2(^5)</td>
</tr>
<tr>
<td>11110</td>
<td>11110</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11111</td>
<td>01011</td>
<td>1:65536</td>
<td>2(^{16})</td>
</tr>
</tbody>
</table>
REGISTER 5-6:  CONFIGURATION WORD 3H (30 0005h)

<table>
<thead>
<tr>
<th>bit 7-6</th>
<th>bit 5-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>U-1 U-1</td>
<td>R/W-1 R/W-1 R/W-1 R/W-1</td>
</tr>
<tr>
<td>WDTCCS&lt;2:0&gt;</td>
<td>WDTCWS&lt;2:0&gt;</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>WDTCWS&lt;2:0&gt;</th>
<th>Window at POR</th>
<th>Software Control of Window</th>
<th>Keyed Access Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Value</td>
<td>Window Delay Percent of Time</td>
<td>Window Opening Percent of Time</td>
</tr>
<tr>
<td>000</td>
<td>000</td>
<td>87.5</td>
<td>12.5</td>
</tr>
<tr>
<td>001</td>
<td>010</td>
<td>75</td>
<td>25</td>
</tr>
<tr>
<td>010</td>
<td>010</td>
<td>62.5</td>
<td>37.5</td>
</tr>
<tr>
<td>011</td>
<td>011</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>37.5</td>
<td>62.5</td>
</tr>
<tr>
<td>101</td>
<td>101</td>
<td>25</td>
<td>75</td>
</tr>
<tr>
<td>110</td>
<td>111</td>
<td>n/a</td>
<td>100</td>
</tr>
<tr>
<td>111</td>
<td>111</td>
<td>n/a</td>
<td>100</td>
</tr>
</tbody>
</table>
REGISTER 5-7: CONFIGURATION WORD 4L (30 0006h)

<table>
<thead>
<tr>
<th>R/W-1</th>
<th>U-1</th>
<th>U-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRTAPP (1)</td>
<td>—</td>
<td>—</td>
<td>SAFEN (1)</td>
<td>BBEN (1)</td>
<td>BBSIZE&lt;2:0&gt; (2)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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 (1)
  - 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 (1)
  - 1 = SAF is disabled
  - 0 = SAF is enabled

**bit 3**
- **BBEN**: Boot Block Enable bit (1)
  - 1 = Boot Block disabled
  - 0 = Boot Block enabled

**bit 2-0**
- **BBSIZE<2:0>**: Boot Block Size Selection bits (2)
  - 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.

**Note 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**

<table>
<thead>
<tr>
<th>BBEN</th>
<th>BBSIZE&lt;2:0&gt;</th>
<th>Boot Block Size (words)</th>
<th>END_ADDRESS_BOOT</th>
<th>Device Size(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16k</td>
</tr>
<tr>
<td>1</td>
<td>xxx</td>
<td>0</td>
<td>—</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>111</td>
<td>512</td>
<td>00 03FFh</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>110</td>
<td>1024</td>
<td>00 07FFh</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>101</td>
<td>2048</td>
<td>00 0FFFh</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>100</td>
<td>4096</td>
<td>00 1FFFh</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>011</td>
<td>8192</td>
<td>00 3FFFh</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>010</td>
<td>16384</td>
<td>00 7FFFh</td>
<td>—</td>
</tr>
<tr>
<td>0</td>
<td>001</td>
<td>32768</td>
<td>00 FFFFh</td>
<td>—</td>
</tr>
<tr>
<td>0</td>
<td>000</td>
<td>32768</td>
<td>00 FFFFh</td>
<td>—</td>
</tr>
</tbody>
</table>

**Note 1:** For each device, the quoted device size specification is listed in Table 4-1.

**Note 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)

<table>
<thead>
<tr>
<th>U-1</th>
<th>U-1</th>
<th>R/W-1</th>
<th>U-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>—</td>
<td>LVP(2)</td>
<td>—</td>
<td>WRTSAF(1,3)</td>
<td>WRTD(1,4)</td>
<td>WRTC(1)</td>
<td>WRTB(1,5)</td>
</tr>
</tbody>
</table>

#### 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

#### bit 5: LVP
- **LVP** = Low-Voltage Programming Enable bit(2)
- 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

#### bit 3: WRTSAF
- **WRTSAF** = Storage Area Flash (SAF) Write Protection bit(1,3)
- 1 = SAF is NOT write-protected
- 0 = SAF is write-protected

#### bit 2: WRTD
- **WRTD** = Data EEPROM Write Protection bit(1,4)
- 1 = Data EEPROM NOT write-protected
- 0 = Data EEPROM write-protected

#### bit 1: WRTC
- **WRTC** = Configuration Register Write Protection bit(1)
- 1 = Configuration Register NOT write-protected
- 0 = Configuration Register write-protected

#### bit 0: WRTB
- **WRTB** = Boot Block Write Protection bit(1,5)
- 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 SAFEN = 0.
- 4: Unimplemented if data EEPROM is not present.
- 5: Only applicable if BBEN = 0.
REGISTER 5-9:  CONFIGURATION WORD 5L (30 0008h)

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>U-1</td>
<td>U-1</td>
</tr>
<tr>
<td>U-1</td>
<td>U-1</td>
</tr>
<tr>
<td>U-1</td>
<td>U-1</td>
</tr>
<tr>
<td>R/W-1</td>
<td>CP</td>
</tr>
</tbody>
</table>

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 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)

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>U-1</td>
<td>U-1</td>
</tr>
<tr>
<td>U-1</td>
<td>U-1</td>
</tr>
<tr>
<td>U-1</td>
<td>U-1</td>
</tr>
<tr>
<td>U-1</td>
<td>U-1</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Default/ Unprogrammed Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>30 0000h</td>
<td>CONFIG1L</td>
<td>—</td>
<td>RSTOSC&lt;2:0&gt;</td>
<td>—</td>
<td>—</td>
<td>FEXTOSC&lt;2:0&gt;</td>
<td>1111 1111</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30 0001h</td>
<td>CONFIG1H</td>
<td>—</td>
<td>—</td>
<td>FCMEN</td>
<td>—</td>
<td>CSWEN</td>
<td>PR1WAY</td>
<td>CLKOUTEN</td>
<td>1111 1111</td>
<td></td>
</tr>
<tr>
<td>30 0002h</td>
<td>CONFIG2L</td>
<td>—</td>
<td>BOREN&lt;1:0&gt;</td>
<td>LBOREN</td>
<td>IVT1WAY</td>
<td>MVECEN</td>
<td>PWRTS&lt;1:0&gt;</td>
<td>MCLRE</td>
<td>1111 1111</td>
<td></td>
</tr>
<tr>
<td>30 0003h</td>
<td>CONFIG2H</td>
<td>XINST</td>
<td>—</td>
<td>DEBUG</td>
<td>STVREN</td>
<td>PPP1WAY</td>
<td>ZCD</td>
<td>BORV&lt;1:0&gt;</td>
<td>1111 1111</td>
<td></td>
</tr>
<tr>
<td>30 0004h</td>
<td>CONFIG3L</td>
<td>—</td>
<td>—</td>
<td>WDTEN&lt;1:0&gt;</td>
<td>—</td>
<td>WDTCP&lt;4:0&gt;</td>
<td>1111 1111</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30 0005h</td>
<td>CONFIG3H</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>WDTCCS&lt;2:0&gt;</td>
<td>WDCWS&lt;2:0&gt;</td>
<td>1111 1111</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30 0006h</td>
<td>CONFIG4L</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>SAFEN</td>
<td>BBEN</td>
<td>BBSIZE&lt;2:0&gt;</td>
<td>1111 1111</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30 0007h</td>
<td>CONFIG4H</td>
<td>—</td>
<td>—</td>
<td>LVP</td>
<td>WRTSAF</td>
<td>WRTD</td>
<td>WRTC</td>
<td>CP</td>
<td>1111 1111</td>
<td></td>
</tr>
<tr>
<td>30 0008h</td>
<td>CONFIG5L</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>1111 1111</td>
<td></td>
</tr>
<tr>
<td>30 0009h</td>
<td>CONFIG5H</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>1111 1111</td>
<td></td>
</tr>
</tbody>
</table>
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 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)F26/27/45/46/47/55/56/57K42 Memory Programming Specification” (DS40001886).
5.5 Device ID and Revision ID

The 16-bit device ID word is located at 3F FFFEh 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

<table>
<thead>
<tr>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>bit 8</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>bit 0</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Device</th>
<th>Device ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIC18F26K42</td>
<td>6C60h</td>
</tr>
<tr>
<td>PIC18F27K42</td>
<td>6C40h</td>
</tr>
<tr>
<td>PIC18F45K42</td>
<td>6C20h</td>
</tr>
<tr>
<td>PIC18F46K42</td>
<td>6C00h</td>
</tr>
<tr>
<td>PIC18F47K42</td>
<td>6BE0h</td>
</tr>
<tr>
<td>PIC18F55K42</td>
<td>6BC0h</td>
</tr>
<tr>
<td>PIC18F56K42</td>
<td>6BA0h</td>
</tr>
<tr>
<td>PIC18F57K42</td>
<td>6B80h</td>
</tr>
<tr>
<td>PIC18LF26K42</td>
<td>6DA0h</td>
</tr>
<tr>
<td>PIC18LF27K42</td>
<td>6D80h</td>
</tr>
<tr>
<td>PIC18LF45K42</td>
<td>6D60h</td>
</tr>
<tr>
<td>PIC18LF46K42</td>
<td>6D40h</td>
</tr>
<tr>
<td>PIC18LF47K42</td>
<td>6D20h</td>
</tr>
<tr>
<td>PIC18LF55K42</td>
<td>6D00h</td>
</tr>
<tr>
<td>PIC18LF56K42</td>
<td>6CE0h</td>
</tr>
<tr>
<td>PIC18LF57K42</td>
<td>6CC0h</td>
</tr>
</tbody>
</table>
## REGISTER 5-12: REVISION ID: REVISION ID REGISTER

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>R</td>
<td>Read as ‘1010’</td>
</tr>
<tr>
<td>14</td>
<td>MJRREV&lt;5:2&gt;</td>
<td>Major Revision ID bits</td>
</tr>
<tr>
<td>8</td>
<td>bit 8</td>
<td>These bits are fixed with value ‘1010’ for all devices in this family.</td>
</tr>
<tr>
<td>7</td>
<td>MNRREV&lt;5:0&gt;</td>
<td>Minor Revision ID bits</td>
</tr>
<tr>
<td>11</td>
<td>bit 11</td>
<td>These bits are used to identify a major revision. A major revision is indicated by revision (A0, B0, C0, etc.)</td>
</tr>
<tr>
<td>6</td>
<td>bit 12</td>
<td>Revision A = 0b00 0000</td>
</tr>
</tbody>
</table>

Legend:
- R = Readable bit
- ‘1’ = Bit is set
- 0’ = Bit is cleared
- x = Bit is unknown
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)F26/27/45/46/47/55/56/57K42 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>
<thead>
<tr>
<th>Address Range</th>
<th>Name of Region</th>
<th>Standard Device Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>3F0000h-3F000Bh</td>
<td>MUI0</td>
<td>Microchip Unique Identifier (6 Words)</td>
</tr>
<tr>
<td></td>
<td>MUI1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MUI2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MUI3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MUI4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MUI5</td>
<td></td>
</tr>
<tr>
<td>3F0000Ch-3F000Fh</td>
<td>MUI6</td>
<td>Unassigned (2 Words)</td>
</tr>
<tr>
<td></td>
<td>MUI7</td>
<td></td>
</tr>
<tr>
<td>3F0010h-3F0023h</td>
<td>EUI0</td>
<td>Optional External Unique Identifier (10 Words)</td>
</tr>
<tr>
<td></td>
<td>EUI1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EUI2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EUI3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EUI4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EUI5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EUI6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EUI7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EUI8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EUI9</td>
<td></td>
</tr>
<tr>
<td>3F0024h-3F0025h</td>
<td></td>
<td>Reserved (1 Word)</td>
</tr>
<tr>
<td>3F0026h-3F0027h</td>
<td>TSLR2</td>
<td>Temperature Indicator ADC reading at @ 90°C (low range setting)</td>
</tr>
<tr>
<td>3F0028h-3F0029h</td>
<td></td>
<td>Reserved (1 Word)</td>
</tr>
<tr>
<td>3F002Ah-3F002Bh</td>
<td></td>
<td>Reserved (1 Word)</td>
</tr>
<tr>
<td>3F002Ch-3F002Dh</td>
<td>TSHR2</td>
<td>Temperature Indicator ADC reading at @ 90°C (high range setting)</td>
</tr>
<tr>
<td>3F002 Eh-3F002Fh</td>
<td></td>
<td>Reserved (1 Word)</td>
</tr>
<tr>
<td>3F0030h-3F0031h</td>
<td>FVRA1X</td>
<td>ADC FVR1 Output voltage for 1x setting (in mV)</td>
</tr>
<tr>
<td>3F0032h-3F0033h</td>
<td>FVRA2X</td>
<td>ADC FVR1 Output Voltage for 2x setting (in mV)</td>
</tr>
<tr>
<td>3F0034h-3F0035h</td>
<td>FVRA4X</td>
<td>ADC FVR1 Output Voltage for 4x setting (in mV)</td>
</tr>
<tr>
<td>3F0036h-3F0037h</td>
<td>FVRC1X</td>
<td>Comparator FVR2 output voltage for 1x setting (in mV)</td>
</tr>
<tr>
<td>3F0038h-3F0039h</td>
<td>FVRC2X</td>
<td>Comparator FVR2 output voltage for 2x setting (in mV)</td>
</tr>
<tr>
<td>3F003Ah-3F003Bh</td>
<td>FVRC4X(1)</td>
<td>Comparator FVR2 output voltage for 4x setting (in mV)</td>
</tr>
<tr>
<td>3F003Ch-3F003Fh</td>
<td></td>
<td>Unassigned (2 Words)</td>
</tr>
</tbody>
</table>

Note 1: Value not present on LF devices.
5.7.1  MICROCHIP UNIQUE IDENTIFIER (MUI)
The PIC18(L)F26/27/45/46/47/55/56/57K42 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™ 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 35.0 “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 35.0 “Temperature Indicator Module”.

- **TSLR2**: Address 3F0026h to 3F0027h store the measurements for the low-range setting of the Temperature Sensor at VDD = 3V.
- **TSHR2**: Address 3F002Ch to 3F002Dh store the measurements for the High Range setting of the Temperature Sensor at VDD = 3V.

The stored measurements are made by the device ADC using the internal VREF = 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 34.0 “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)F26/27/45/55/46/47/56/57K42 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>
<thead>
<tr>
<th>ADDRESS</th>
<th>Name</th>
<th>DESCRIPTION</th>
<th>VALUE PIC18(L)F26/27/45/55/46/47/56/57K42</th>
<th>UNITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>3F FF00h-3F FF01h</td>
<td>ERSIZ</td>
<td>Erase Row Size</td>
<td>64</td>
<td>Words</td>
</tr>
<tr>
<td>3F FF02h-3F FF03h</td>
<td>WLSIZ</td>
<td>Number of write latches per row</td>
<td>128</td>
<td>Bytes</td>
</tr>
<tr>
<td>3F FF04h-3F FF05h</td>
<td>URSIZ</td>
<td>Number of User Rows</td>
<td>256</td>
<td>Rows</td>
</tr>
<tr>
<td>3F FF06h-3F FF07h</td>
<td>EESIZ</td>
<td>Data EEPROM memory size</td>
<td>256</td>
<td>Bytes</td>
</tr>
<tr>
<td>3F FF08h-3F FF09h</td>
<td>PCNT</td>
<td>Pin Count</td>
<td>40(1)×48</td>
<td>Pins</td>
</tr>
</tbody>
</table>

Note 1: Pin count of 40 is also used for 44-pin part.
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 (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 6-2: LPBOR, BOR, POR RELATIONSHIP

- REARM POR Event
- POR
- POR Event
- BOR BOR Event
- LPBOR LPBOR Event
- To PCON0 indicator bit
- Reset logic
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 44-13 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

<table>
<thead>
<tr>
<th>BOREN&lt;1:0&gt;</th>
<th>SBOREN</th>
<th>Device Mode</th>
<th>BOR Mode</th>
<th>Instruction Execution upon:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Release of POR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Active Wait for release of BOR (BORRDY = 1)</td>
</tr>
<tr>
<td>11</td>
<td>X</td>
<td>X</td>
<td>Active</td>
<td>Wait for release of BOR (BORRDY = 1)</td>
</tr>
<tr>
<td>10</td>
<td>X</td>
<td>Awake</td>
<td>Active</td>
<td>Wait for release of BOR (BORRDY = 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sleep</td>
<td>Hibernate</td>
<td>Wait for release of BOR (BORRDY = 1)</td>
</tr>
<tr>
<td>01</td>
<td>1</td>
<td>X</td>
<td>Active</td>
<td>Wait for release of BOR (BORRDY = 1)</td>
</tr>
<tr>
<td>00</td>
<td>X</td>
<td>X</td>
<td>Disabled</td>
<td></td>
</tr>
</tbody>
</table>

FIGURE 6-3: BROWN-OUT SITUATIONS

Note 1: TPWRT delay depends on PWRTS<1:0> Configuration bits.
### 6.3 Register Definitions: BOR Control

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

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value at POR and BOR/Value at all other Resets</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>SBOREN: Software Brown-out Reset Enable bit</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td><strong>If BOREN ≠ 01:</strong></td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>SBOREN is read/write, but has no effect on the BOR.</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td><strong>If BOREN = 01:</strong></td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>1 = BOR Enabled</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0 = BOR Disabled</td>
<td>-</td>
</tr>
<tr>
<td>6-1</td>
<td>Unimplemented</td>
<td>-</td>
</tr>
<tr>
<td>0</td>
<td>BORRDY: Brown-out Reset Circuit Ready Status bit</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>1 = The Brown-out Reset Circuit is active and armed</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0 = The Brown-out Reset Circuit is disabled or is warming up</td>
<td>-</td>
</tr>
</tbody>
</table>

**Legend:**

- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as ‘0’
- **u** = Bit is unchanged
- **x** = Bit is unknown
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- ‘-n/n’ = Value at POR and BOR/Value at all other Resets
- **q** = Value depends on condition

---

### Notes:

- **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

- **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 LPBOREN bit of Configuration Word 2L. When the device is erased, the LPBOR module defaults to disabled.

6.5 MCLR

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

<table>
<thead>
<tr>
<th>MCLRE</th>
<th>LVP</th>
<th>MCLR</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>1</td>
<td>Enabled</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Enabled</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>Disabled</td>
</tr>
</tbody>
</table>

6.5.1 MCLR ENABLED

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

The device has a noise filter in the MCLR Reset path. The filter will detect and ignore small pulses.

Note: An internal Reset event (RESET instruction, BOR, WWDT, POR stack), does not drive the MCLR pin low.

6.5.2 MCLR DISABLED

When MCLR is disabled, the MCLR pin becomes input-only and pin functions such as internal weak pull-ups are under software control. See Section 16.1 “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 CLRWD T instruction within the time-out period or window set. The TO and PD bits in the STATUS register and the RWD T 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 RESET Instruction

A RESET instruction will cause a device Reset. The R bit in the PCON0 register will be set to '0'. See Table 6-3 for default conditions after a 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 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. 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 MCLR Reset. If MCLR is kept low long enough, the Power-up Timer and oscillator Start-up Timer will expire. Upon bringing MCLR high, the device will begin execution after 10 FOSC 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**

![Reset Start-up Sequence Diagram]
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>
<thead>
<tr>
<th>Condition</th>
<th>Program Counter</th>
<th>STATUS Register(^{1,2})</th>
<th>PCON0 Register</th>
<th>PCON1 Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power-on Reset</td>
<td>0</td>
<td>-110 0000</td>
<td>0011 110x</td>
<td>---- --1-</td>
</tr>
<tr>
<td>Brown-out Reset</td>
<td>0</td>
<td>-110 0000</td>
<td>0011 11u0</td>
<td>---- --1-</td>
</tr>
<tr>
<td>MCLR Reset during normal operation</td>
<td>0</td>
<td>-uuu uuuu</td>
<td>uuuu 0uuu</td>
<td>---- --u-</td>
</tr>
<tr>
<td>MCLR Reset during Sleep</td>
<td>0</td>
<td>-10u uuuu</td>
<td>uuuu 0uuu</td>
<td>---- --u-</td>
</tr>
<tr>
<td>WWDT Time-out Reset</td>
<td>0</td>
<td>-0uu uuuu</td>
<td>uuuu 0uuu</td>
<td>---- --u-</td>
</tr>
<tr>
<td>WWDT Window Violation Reset</td>
<td>0</td>
<td>-uuu uuuu</td>
<td>uuu0 uuuu</td>
<td>---- --u-</td>
</tr>
<tr>
<td>RESET Instruction Executed</td>
<td>0</td>
<td>-uuu uuuu</td>
<td>uuuu u0uu</td>
<td>---- --u-</td>
</tr>
<tr>
<td>Stack Overflow Reset (STVREN = 1)</td>
<td>0</td>
<td>-uuu uuuu</td>
<td>1uuu uuuu</td>
<td>---- --u-</td>
</tr>
<tr>
<td>Stack Underflow Reset (STVREN = 1)</td>
<td>0</td>
<td>-uuu uuuu</td>
<td>uluu uuuu</td>
<td>---- --u-</td>
</tr>
<tr>
<td>Memory Violation Reset</td>
<td>0</td>
<td>-uuu uuuu</td>
<td>uuuu uuuu</td>
<td>---- --0-</td>
</tr>
</tbody>
</table>

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 (BOR)
- Power-on Reset (POR)
- Reset Instruction Reset (RI)
- MCLR Reset (RMCLR)
- Watchdog Timer Reset (RWDT)
- Watchdog Window Violation (WDTWV)
- Stack Underflow Reset (STKUNF)
- Stack Overflow Reset (STKOVF)
- Memory Violation Reset (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

<table>
<thead>
<tr>
<th>R/W/HS-0/q</th>
<th>R/W/HS-0/q</th>
<th>R/W/HC-1/q</th>
<th>R/W/HC-1/q</th>
<th>R/W/HC-1/q</th>
<th>R/W/HC-1/q</th>
<th>R/W/HC-0/u</th>
<th>R/W/HC-q/u</th>
</tr>
</thead>
<tbody>
<tr>
<td>STKOVF</td>
<td>STKUNF</td>
<td>WDTWV</td>
<td>RWDT</td>
<td>RMCLR</td>
<td>RI</td>
<td>POR</td>
<td>BOR</td>
</tr>
</tbody>
</table>

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 CLRWDTR 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

| 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>BORCON</td>
<td>SBOREN</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>BORRDY 85</td>
</tr>
<tr>
<td>PCON0</td>
<td>STKOVF</td>
<td>STKUNF</td>
<td>WDWTWV</td>
<td>RWDT</td>
<td>RMCLR</td>
<td>RI</td>
<td>POR</td>
<td>BOR</td>
<td>90</td>
</tr>
<tr>
<td>PCON1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>MEMV</td>
<td>91</td>
</tr>
</tbody>
</table>

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 (between 100 kHz and 8 MHz)
6. HS – High Gain Crystal or Ceramic Resonator mode (above 4 MHz)

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

- CLKIN/OSC1
- External Oscillator (EXTOSC)
- CLKOUT/OSC2
- Secondary Oscillator (SOSC)
- SOSCIN/SOSCI
- SOSCO
- LFINTOSC: 31 kHz Oscillator
- HFINTOSC: HFFRQ<3:0> 1.2,4,8,12,16,32,48,64 MHz Oscillator
- MFINTOSC: 31.25 kHz and 500 kHz Oscillator
- COSC<2:0>
- CDIV<4:0>
- 4x PLL
- Post Divider
- LFINTOSC is used to monitor system clock
- FSCM
- System Clock
- Peripheral Clock
- SYSCMD
- Sleep
- Idle
- To Peripherals
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 HFINTOSC 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

Refer to Table 44-9 for External Clock/Oscillator Timing Requirements. 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 designed to drive only 32.768 kHz tuning-fork type crystals (watch crystals).

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 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.
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 44-11.

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).

Note 1: A series resistor (RS) may be required for quartz crystals with low drive level.
2: The value of RF varies with the Oscillator mode selected (typically between 2 MΩ to 10 MΩ).
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 SOSCI and SOSCO device pins, or an external clock source connected to the SOSCIN pin. The secondary oscillator can be selected during runtime 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 by setting the FRQ<3:0> bits of the OSCFRQ 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 ADC² 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 ADC² 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)

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.

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”.

© 2017 Microchip Technology Inc. Preliminary

DS40001919B-page 99
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)

<table>
<thead>
<tr>
<th>OSCCON1 WRITTEN</th>
<th>OSC #1</th>
<th>OSC #2</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORDY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOSCR</td>
<td></td>
<td>NOTE 2</td>
</tr>
<tr>
<td>CSWIF</td>
<td></td>
<td>NOTE 1</td>
</tr>
<tr>
<td>CSWHOLD</td>
<td>USER CLEAR</td>
<td></td>
</tr>
</tbody>
</table>

**Note 1:** CSWIF is asserted coincident with NOSCR; interrupt is serviced at OSC#2 speed.

**Note 2:** The assertion of NOSCR is hidden from the user because it appears only for the duration of the switch.

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

<table>
<thead>
<tr>
<th>OSCCON1 WRITTEN</th>
<th>OSC #1</th>
<th>OSC #2</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORDY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOSCR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSWIF</td>
<td></td>
<td>NOTE 1</td>
</tr>
<tr>
<td>CSWHOLD</td>
<td>USER CLEAR</td>
<td></td>
</tr>
</tbody>
</table>

**Note 1:** CSWIF is asserted coincident with NOSCR, and may be cleared before or after clearing CSWHOLD = 0.
FIGURE 7-8: CLOCK SWITCH ABANDONED

Note 1: CSWIF may be cleared before or after rewriting OSCCON1; CSWIF is not automatically cleared.

2: ORDY = 0 if OSCCON1 does not match OSCCON2; a new switch will begin.
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).

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

<table>
<thead>
<tr>
<th>NOSC&lt;2:0&gt;</th>
<th>COSC&lt;2:0&gt;</th>
<th>Clock Source</th>
<th>Clock Divider</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td></td>
<td>EXTOSC&lt;1&gt;</td>
<td>1111-1010</td>
</tr>
<tr>
<td>110</td>
<td></td>
<td>HFINTOSC&lt;2&gt;</td>
<td>1001</td>
</tr>
<tr>
<td>101</td>
<td></td>
<td>LFINTOSC</td>
<td>1000</td>
</tr>
<tr>
<td>100</td>
<td></td>
<td>SOSC</td>
<td>0111</td>
</tr>
<tr>
<td>011</td>
<td></td>
<td>Reserved</td>
<td>0110</td>
</tr>
<tr>
<td>010</td>
<td></td>
<td>EXTOSC + 4x PLL&lt;3&gt;</td>
<td>0101</td>
</tr>
<tr>
<td>001</td>
<td></td>
<td>Reserved</td>
<td>0100</td>
</tr>
<tr>
<td>000</td>
<td></td>
<td>Reserved</td>
<td>0011</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0010</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0001</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0000</td>
</tr>
</tbody>
</table>

Note 1:  EXTOSC configured by the FEXTOSC bits of Configuration Word 1 (Register 5-1).
2:  HFINTOSC frequency is set with the FRQ bits of the OSCFRQ register (Register 7-5).
3:  EXTOSC must meet the PLL specifications (Table 44-11).
7.5 Register Definitions: Oscillator Control

REGISTER 7-1: OSCCON1: OSCILLATOR CONTROL REGISTER 1

<table>
<thead>
<tr>
<th>U-0</th>
<th>R/W-ff</th>
<th>R/W-ff</th>
<th>R/W-ff</th>
<th>R/W-q/q</th>
<th>R/W-q/q</th>
<th>R/W-q/q</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NOSC&lt;2:0&gt;</td>
<td>NDIV&lt;3:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as ‘0’
u = Bit is unchanged
x = Bit is unknown
’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\(^{(1,2,3)}\)
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\(^{(2,3)}\)
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

<table>
<thead>
<tr>
<th>RSTOSC</th>
<th>SFR Reset Values</th>
<th>Initial Fosc Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NOSC/COSC</td>
<td>CDIV</td>
</tr>
<tr>
<td>111</td>
<td>111</td>
<td>1:1</td>
</tr>
<tr>
<td>110</td>
<td>110</td>
<td>4:1</td>
</tr>
<tr>
<td>101</td>
<td>101</td>
<td>1:1</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>1:1</td>
</tr>
<tr>
<td>011</td>
<td></td>
<td></td>
</tr>
<tr>
<td>010</td>
<td>010</td>
<td>1:1</td>
</tr>
<tr>
<td>001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000</td>
<td>110</td>
<td>1:1</td>
</tr>
</tbody>
</table>

Note 1: EXTOSC must meet the PLL specifications (Table 44-11).
REGISTER 7-2:  OSCCON2: OSCILLATOR CONTROL REGISTER 2

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 6-4</th>
<th>bit 3-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unimplemented: Read as '0'</td>
<td>COSC&lt;2:0&gt;: Current Oscillator Source Select bits (read-only)(^{(1)})</td>
<td>CDIV&lt;3:0&gt;: Current Divider Select bits (read-only)(^{(1)})</td>
</tr>
</tbody>
</table>

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

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

REGISTER 7-3:  OSCCON3: OSCILLATOR CONTROL REGISTER 3

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 6-5</th>
<th>bit 4-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSWHOLD: Clock Switch Hold bit</td>
<td>SOSCPWR: Secondary Oscillator Power Mode Select bit</td>
<td>ORDY: Oscillator Ready bit (read-only)</td>
</tr>
<tr>
<td>1 = Clock switch will hold (with interrupt) when the oscillator selected by NOSC is ready</td>
<td>1 = Secondary oscillator operating in High-Power mode</td>
<td>1 = OSCCON1 = OSCCON2; the current system clock is the clock specified by NOSC</td>
</tr>
<tr>
<td>0 = Clock switch may proceed when the oscillator selected by NOSC is ready; NOSCR becomes '1', the switch will occur</td>
<td>0 = Secondary oscillator operating in Low-Power mode</td>
<td>0 = A clock switch is in progress</td>
</tr>
</tbody>
</table>

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

**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

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXTOR</td>
<td>PLLR</td>
</tr>
<tr>
<td>HFOR</td>
<td></td>
</tr>
<tr>
<td>MFOR</td>
<td></td>
</tr>
<tr>
<td>LFOR</td>
<td></td>
</tr>
<tr>
<td>SOR</td>
<td></td>
</tr>
<tr>
<td>ADOR</td>
<td>—</td>
</tr>
</tbody>
</table>

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 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 ready to be used

- **bit 5 MFOR:** MFINTOSC Oscillator Ready
  - 1 = The oscillator is ready to be used
  - 0 = The oscillator is not enabled, or is not 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 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 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 ready to be used

- **bit 1 Unimplemented:** Read as ‘0’

- **bit 0 PLLR:** 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

<table>
<thead>
<tr>
<th></th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-q/q</th>
<th>R/W-q/q</th>
<th>R/W-q/q</th>
<th>R/W-q/q</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
</tbody>
</table>

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

### bit 7-4
**Unimplemented:** Read as ‘0’

### bit 3-0
**FRQ<3:0>: HFINTOSC Frequency Selection bits**

<table>
<thead>
<tr>
<th>FRQ&lt;3:0&gt;</th>
<th>Nominal Freq (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td></td>
</tr>
<tr>
<td>1010</td>
<td></td>
</tr>
<tr>
<td>1111</td>
<td></td>
</tr>
<tr>
<td>1110</td>
<td>Reserved</td>
</tr>
<tr>
<td>1101</td>
<td></td>
</tr>
<tr>
<td>1100</td>
<td></td>
</tr>
<tr>
<td>1011</td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>64</td>
</tr>
<tr>
<td>0111</td>
<td>48</td>
</tr>
<tr>
<td>0110</td>
<td>32</td>
</tr>
<tr>
<td>0101</td>
<td>16</td>
</tr>
<tr>
<td>0100</td>
<td>12</td>
</tr>
<tr>
<td>0011</td>
<td>8</td>
</tr>
<tr>
<td>0010</td>
<td>4</td>
</tr>
<tr>
<td>0001</td>
<td>2</td>
</tr>
<tr>
<td>0000</td>
<td>1</td>
</tr>
</tbody>
</table>

**Note 1:** Refer to Table 7-2 for more information.
REGISTER 7-6: OSCTUNE: HFINTOSC TUNING REGISTER

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

— — TUN<5:0>

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  TUN<5:0>: HFINTOSC Frequency Tuning bits

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>01 1111</td>
<td>Maximum frequency</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>00 0000</td>
<td>Center frequency. Oscillator module is running at the calibrated frequency (default value).</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>10 0000</td>
<td>Minimum frequency</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## REGISTER 7-7: OSCEN: OSCILLATOR MANUAL ENABLE REGISTER

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td><strong>EXTOEN</strong>: External Oscillator Manual Request Enable bit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = EXTO SC is explicitly enabled, operating as specified by FEXTOSC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = EXTO SC could be enabled by requesting peripheral</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td><strong>HFOEN</strong>: HFINTOSC Oscillator Manual Request Enable bit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = HFINTOSC is explicitly enabled, operating as specified by OSCFRQ (Register 7-5)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = HFINTOSC could be enabled by requesting peripheral</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td><strong>MFOEN</strong>: MFINTOSC (500 kHz/31.25 kHz) Oscillator Manual Request Enable bit (Derived from HFINTOSC)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = MFINTOSC is explicitly enabled</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = MFINTOSC could be enabled by requesting peripheral</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td><strong>LFOEN</strong>: LFINTOSC (31 kHz) Oscillator Manual Request Enable bit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = LFINTOSC is explicitly enabled</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = LFINTOSC could be enabled by requesting peripheral</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td><strong>SOSCE N</strong>: Secondary Oscillator Manual Request Enable bit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = Secondary Oscillator is explicitly enabled, operating as specified by SOSCPWR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = Secondary Oscillator could be enabled by requesting peripheral</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td><strong>ADOEN</strong>: ADC Oscillator Manual Request Enable bit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = ADC oscillator is explicitly enabled</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = ADC oscillator could be enabled by requesting peripheral</td>
<td></td>
</tr>
<tr>
<td>1-0</td>
<td>Unimplemented: Read as '0'</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSCCON1</td>
<td>—</td>
<td></td>
<td>NOSC&lt;2:0&gt;</td>
<td></td>
<td></td>
<td>NDIV&lt;3:0&gt;</td>
<td></td>
<td></td>
<td>104</td>
</tr>
<tr>
<td>OSCCON2</td>
<td>—</td>
<td></td>
<td>COSC&lt;2:0&gt;</td>
<td></td>
<td></td>
<td>CDIV&lt;3:0&gt;</td>
<td></td>
<td></td>
<td>105</td>
</tr>
<tr>
<td>OSCCON3</td>
<td>CSWHOLD</td>
<td>SOSCPWR</td>
<td>—</td>
<td>ORDY</td>
<td>NOSCR</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>105</td>
</tr>
<tr>
<td>OSCSTAT</td>
<td>EXTOR</td>
<td></td>
<td>HFOR</td>
<td>MFOR</td>
<td>LFOR</td>
<td>SOR</td>
<td>ADOR</td>
<td>—</td>
<td>PLLR 106</td>
</tr>
<tr>
<td>OSCTUNE</td>
<td>—</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TUN&lt;5:0&gt;</td>
<td>108</td>
</tr>
<tr>
<td>OSCFRQ</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>FRQ&lt;3:0&gt; 107</td>
</tr>
<tr>
<td>OSCEN</td>
<td>EXTOEN</td>
<td>HFOEN</td>
<td>MFOEN</td>
<td>LFOEN</td>
<td>SOSCEN</td>
<td>ADOEN</td>
<td>—</td>
<td>—</td>
<td>109</td>
</tr>
</tbody>
</table>

*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).

Note: 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.

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>Bit Name Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLKR</td>
<td>CLKR</td>
</tr>
</tbody>
</table>

REGISTER 8-1: CLKRCON: REFERENCE CLOCK CONTROL REGISTER

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>EN: Reference Clock Module Enable bit</td>
</tr>
<tr>
<td>6-5</td>
<td>Unimplemented: Read as '0'</td>
</tr>
<tr>
<td>4-3</td>
<td>DC&lt;1:0&gt;: Reference Clock Duty Cycle bits(1)</td>
</tr>
<tr>
<td>2-0</td>
<td>DIV&lt;2:0&gt;: Reference Clock Divider bits</td>
</tr>
</tbody>
</table>

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(1)
   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.
REGISTER 8-2:  CLKRCLK: CLOCK REFERENCE CLOCK SELECTION MUX

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>CLK&lt;3:0&gt;</td>
</tr>
</tbody>
</table>

Legend:
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as ‘0’
- u = Bit is unchanged
- x = Bit is unknown
- ‘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

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLKRCON</td>
<td>EN</td>
<td>—</td>
<td>—</td>
<td>DC&lt;1:0&gt;</td>
<td></td>
<td>DIV&lt;2:0&gt;</td>
<td></td>
<td></td>
<td>113</td>
</tr>
<tr>
<td>CLKRCLK</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>114</td>
</tr>
</tbody>
</table>

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 Register 9-36, Register 9-37 and Register 9-38 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 form IVTBASE to (IVTBASE +2n-1) can be encompassed inside the program flash memory.

For example, the K42 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 << 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*Vector Number).

This calculated Interrupt Vector Address value is stored in the IVTAD<20:0> registers when an interrupt is received (Registers 9-39 through 9-41).

User-assigned software priority assigned using the IPRx registers does not affect address calculation and is only used to resolve concurrent interrupts.

Note: It is required that the user assign an even address to the IVTBASE register for correct operation.

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).
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.

**TABLE 9-1: IVT ADDRESS CALCULATION SUMMARY**

| IVT Address Calculation | Interrupt Priority
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>INTCON0 Register, IPEN bit</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Multi-Vector Enable</td>
<td>IVTBASE</td>
</tr>
<tr>
<td>CONFIG 2L register</td>
<td></td>
</tr>
<tr>
<td>MVECEN bit</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>IVTBASE + 2^*(Vector Number)</td>
</tr>
</tbody>
</table>

The user must follow the sequence shown in Example 9-2 to set the IVTLOCKED bit.

**EXAMPLE 9-2: IVT LOCK SEQUENCE**

```assembly
; Disable Interrupts:
BCF INTCON0, GIE;

; Bank to IVTLOCK register
BANKSEL IVTLOCK;
MOVLW 55h;

; Required sequence, next 4 instructions
MOVWF IVTLOCK;
MOVWF IVTLOCK;
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.

**EXAMPLE 9-1: IVT UNLOCK SEQUENCE**

```assembly
; Disable Interrupts:
BCF INTCON0, GIE;

; Bank to IVTLOCK register
BANKSEL IVTLOCK;
MOVLW 55h;

; Required sequence, next 4 instructions
MOVWF IVTLOCK;
MOVWF IVTLOCK;
MOVWF IVTLOCK;

; Clear IVTLOCKED bit to enable writes
BCF IVTLOCK, IVTLOCKED;

; Enable Interrupts
BSF INTCON0, GIE;
```
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 9-25 through 9-35).

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.(1)
   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(2).

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

<table>
<thead>
<tr>
<th>Vector Number</th>
<th>Interrupt Source</th>
<th>Vector Number</th>
<th>Interrupt Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Software Interrupt</td>
<td>42</td>
<td>DMA2SCNT</td>
</tr>
<tr>
<td>1</td>
<td>HLVD</td>
<td>43</td>
<td>DMA2DCNT</td>
</tr>
<tr>
<td>2</td>
<td>OSF</td>
<td>44</td>
<td>DMA2OR</td>
</tr>
<tr>
<td>3</td>
<td>CSW</td>
<td>45</td>
<td>DMA2A</td>
</tr>
<tr>
<td>4</td>
<td>NVM</td>
<td>46</td>
<td>I2C2RX</td>
</tr>
<tr>
<td>5</td>
<td>SCAN</td>
<td>47</td>
<td>I2C2TX</td>
</tr>
<tr>
<td>6</td>
<td>CRC</td>
<td>48</td>
<td>I2C2</td>
</tr>
<tr>
<td>7</td>
<td>IOC</td>
<td>49</td>
<td>I2C2E</td>
</tr>
<tr>
<td>8</td>
<td>INT0</td>
<td>50</td>
<td>U2RX</td>
</tr>
<tr>
<td>9</td>
<td>ZCD</td>
<td>51</td>
<td>U2TX</td>
</tr>
<tr>
<td>10</td>
<td>AD</td>
<td>52</td>
<td>U2E</td>
</tr>
<tr>
<td>11</td>
<td>ADT</td>
<td>53</td>
<td>U2</td>
</tr>
<tr>
<td>12</td>
<td>C1</td>
<td>54</td>
<td>TMR3</td>
</tr>
<tr>
<td>13</td>
<td>SMT1</td>
<td>55</td>
<td>TMR3G</td>
</tr>
<tr>
<td>14</td>
<td>SMT1PRA</td>
<td>56</td>
<td>TMR4</td>
</tr>
<tr>
<td>15</td>
<td>SMT1PWA</td>
<td>57</td>
<td>CCP2</td>
</tr>
<tr>
<td>16</td>
<td>DMA1SCNT</td>
<td>58</td>
<td>—</td>
</tr>
<tr>
<td>17</td>
<td>DMA1DCNT</td>
<td>59</td>
<td>CGW2</td>
</tr>
<tr>
<td>18</td>
<td>DMA1OR</td>
<td>60</td>
<td>CLC2</td>
</tr>
<tr>
<td>19</td>
<td>DMA1A</td>
<td>61</td>
<td>INT2</td>
</tr>
<tr>
<td>20</td>
<td>SPI1RX</td>
<td>62</td>
<td>—</td>
</tr>
<tr>
<td>21</td>
<td>SPI1TX</td>
<td>63</td>
<td>—</td>
</tr>
<tr>
<td>22</td>
<td>SPI1</td>
<td>64</td>
<td>—</td>
</tr>
<tr>
<td>23</td>
<td>I2C1RX</td>
<td>65</td>
<td>—</td>
</tr>
<tr>
<td>24</td>
<td>I2C1TX</td>
<td>66</td>
<td>—</td>
</tr>
<tr>
<td>25</td>
<td>I2C1</td>
<td>67</td>
<td>—</td>
</tr>
<tr>
<td>26</td>
<td>I2C1E</td>
<td>68</td>
<td>—</td>
</tr>
<tr>
<td>27</td>
<td>U1RX</td>
<td>69</td>
<td>—</td>
</tr>
<tr>
<td>28</td>
<td>U1TX</td>
<td>70</td>
<td>TMR5</td>
</tr>
<tr>
<td>29</td>
<td>U1E</td>
<td>71</td>
<td>TMR5G</td>
</tr>
<tr>
<td>30</td>
<td>U1</td>
<td>72</td>
<td>TMR6</td>
</tr>
<tr>
<td>31</td>
<td>TMR0</td>
<td>73</td>
<td>CCP3</td>
</tr>
<tr>
<td>32</td>
<td>TMR1</td>
<td>74</td>
<td>CGW3</td>
</tr>
<tr>
<td>33</td>
<td>TMR1G</td>
<td>75</td>
<td>CLC3</td>
</tr>
<tr>
<td>34</td>
<td>TMR2</td>
<td>76</td>
<td>—</td>
</tr>
<tr>
<td>35</td>
<td>CCP1</td>
<td>77</td>
<td>—</td>
</tr>
<tr>
<td>36</td>
<td>—</td>
<td>78</td>
<td>—</td>
</tr>
<tr>
<td>37</td>
<td>NCO</td>
<td>79</td>
<td>—</td>
</tr>
<tr>
<td>38</td>
<td>CGW1</td>
<td>80</td>
<td>CCP4</td>
</tr>
<tr>
<td>39</td>
<td>CLC1</td>
<td>81</td>
<td>CLC4</td>
</tr>
</tbody>
</table>

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

- **HIGH INTSTAT = 00**: High Interrupt addressed, Low Interrupt pending
- **HIGH INTSTAT = 10**: High Interrupt requested, Low Interrupt pending
- **LOW INTSTAT = 01**: Low Interrupt addressed, High Interrupt pending
- **LOW INTSTAT = 11**: Low Interrupt requested

Legend:
- High Interrupt requested
- Low Interrupt requested
- High Interrupt addressed, No pending interrupts
- Low Interrupt addressed, No pending interrupts
- High Interrupt addressed, Low Interrupt pending
- Low Interrupt addressed, High Interrupt pending
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 serviced halfway before moving on to a high priority ISR, then the low priority ISR is completely serviced even if 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 9-43), 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

- **MAIN INTSTAT = 00**
  - Restore Main context
  - Save Main context

- **HIGH INTSTAT = 10**
  - Save Context
  - No Context Save/Restore

- **LOW INTSTAT = 01**
  - No Context Save/Restore

- **HIGH INTSTAT = 11**
  - Save Low context
  - Restore Low context

- **No Context Save/Restore**
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 illustrate 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

Instruction @ X is a One-cycle Instruction

Interrupt Location = Interrupt vector table entry << 2
= 0x86 << 2 = 0x218

Note 1: Instruction @ X is a One-cycle Instruction
FIGURE 9-8: INTERRUPT TIMING DIAGRAM - TWO WORD INSTRUCTION

Interrupt Location = Interrupt vector table entry << 2
= 0x86 << 2 = 0x218

Note 1: Instruction @ Y is a Two-cycle instruction.
FIGURE 9-9: INTERRUPT TIMING DIAGRAM - THREE CYCLE INSTRUCTION

<table>
<thead>
<tr>
<th>Program Counter</th>
<th>System Clock</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z</td>
<td>Z+2</td>
</tr>
</tbody>
</table>

Instruction Register

- Inst @ Z
- Inst @ Z+2
- Inst @ Z+4
- FNOP
- FNOP
- FNOP
- BCF
- RETFIE

Interrupt Location = Interrupt vector table entry << 2
= 0x86 << 2 = 0x218

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.

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 PIEx 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)F26/27/45/46/47/55/56/57K42 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/GIEL, 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 PIC®16 microcontroller mid-range devices. All interrupts branch to address 0008h since the interrupt priority is disabled.

Note: At a device Reset, the IPRx registers are initialized, such that all user interrupt sources are assigned to high priority.
**EXAMPLE 9-3: SETTING UP VECTORED INTERRUPTS USING MPASM**

```assembly
ISR_TMR0: CODE 0x8C0 ; ISR code at 0x08C0 in PFM
BANKSEL PIR0 ; Select bank for PIR0
BCF PIR3, TMR0IF ; Clear TMR0IF
BTG LATC, 0, ACCESS ; Code to execute in ISR
RETFIE 1 ; Return from ISR

InterruptInit:
BANKSEL INTCON0 ; Select bank for INTCON0
BSF INTCON0, GIEH ; Enable high priority interrupts
BSF INTCON0, GIEL ; Enable low priority interrupts
BSF INTCON0, IPEN_INTCON0 ; Enable interrupt priority

BANKSEL PIE0 ; Select bank for PIE0
BSF PIE3, TMR0IE ; Enable TMR0 interrupt
BSF PIE4, TMR1IE ; Enable TMR1 interrupt

BCF IPR3, TMR0IP ; Make TMR0 interrupt low priority
RETURN 1

VectorTableInit:
; Set IVTBASE (optional - default is 0x000000)
MOVLW 0x00 ; This is optional
MOVWF IVTBASEU, ACCESS ; If not included, then the
MOVLW 0x40 ; hardware default value of
MOVWF IVTBASEH, ACCESS ; 0x0008 will be taken.
MOVLW 0x08
MOVWF IVTBASEL, ACCESS

; TMR0 vector at IVTBASE + 2*(TMR0 vector number i.e. 31) = 0x4046
MOVLW 0x00 ; Load TBLPTR with the
MOVWF TBLPTRU, ACCESS ; PFM memory location to be
MOV LW 0x40 ; written to.
MOVWF TBLPTRH, ACCESS
MOV LW 0x46
MOVWF TBLPTRL, ACCESS

; Write the contents of TMR0 vector location
; ISR_TMR0_ADDRESS >> 2 = 0x08C0 >> 2 = 0x0230
MOVLW 0x30 ; Low byte first
MOVWF TABLAT, ACCESS ; Write to temp table latch
TBLWT++
MOVLW 0x02 ; High byte next
MOVWF TABLAT, ACCESS ; Write to temp table latch
TBLWT++

; Write to PFM now using NVMCON
BANKSEL NVMCON1 ; Select bank for NVMCON1
MOV LW 0x84 ; Setting to write to PFM
MOVWF NVMCON1

MOV LW 0x55 ; Required unlock sequence
MOVWF NVMCON2
MOV LW 0xAA
MOVWF NVMCON2
BSF NVMCON1, WR ; Start writing to PFM
BTFSC NVMCON1, WR ; Wait for write to complete
GOTO $-2
RETURN 1
```
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
    IVBTASEL = 0x08;
}
9.12 Register Definitions: Interrupt Control

REGISTER 9-1: INTCON0: INTERRUPT CONTROL REGISTER 0

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 6</th>
<th>bit 5</th>
<th>bit 4-3</th>
<th>bit 2</th>
<th>bit 1</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>GIE/GIEH</td>
<td>GIEL</td>
<td>IPEN</td>
<td>—</td>
<td>—</td>
<td>INT2EDG</td>
<td>INT1EDG</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>bit 7-0</th>
<th>STAT&lt;1:0&gt;: Interrupt State Status bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>High priority ISR executing, high priority interrupt was received while a low priority ISR was executing</td>
</tr>
<tr>
<td>10</td>
<td>High priority ISR executing, high priority interrupt was received in main routine</td>
</tr>
<tr>
<td>01</td>
<td>Low priority ISR executing, low priority interrupt was received in main routine</td>
</tr>
<tr>
<td>00</td>
<td>Main routine executing</td>
</tr>
</tbody>
</table>

### 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
### REGISTER 9-3: PIR0: PERIPHERAL INTERRUPT REQUEST REGISTER 0

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>IOCIF: Interrupt-on-Change Interrupt Flag bit(2)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 = Interrupt has occurred</td>
</tr>
<tr>
<td></td>
<td>0 = Interrupt event has not occurred</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 6</th>
<th>CRCIF: CRC Interrupt Flag bit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 = Interrupt has occurred (must be cleared by software)</td>
</tr>
<tr>
<td></td>
<td>0 = Interrupt event has not occurred</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 5</th>
<th>SCANIF: Memory Scanner Interrupt Flag bit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 = Interrupt has occurred (must be cleared by software)</td>
</tr>
<tr>
<td></td>
<td>0 = Interrupt event has not occurred</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 4</th>
<th>NVMIF: NVM Interrupt Flag bit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 = Interrupt has occurred (must be cleared by software)</td>
</tr>
<tr>
<td></td>
<td>0 = Interrupt event has not occurred</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 3</th>
<th>CSWIF: Clock Switch Interrupt Flag bit(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 = Interrupt has occurred (must be cleared by software)</td>
</tr>
<tr>
<td></td>
<td>0 = Interrupt event has not occurred</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 2</th>
<th>OSFIF: Oscillator Fail Interrupt Flag bit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 = Interrupt has occurred (must be cleared by software)</td>
</tr>
<tr>
<td></td>
<td>0 = Interrupt event has not occurred</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 1</th>
<th>HLVDIF: HLVD Interrupt Flag bit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 = Interrupt has occurred (must be cleared by software)</td>
</tr>
<tr>
<td></td>
<td>0 = Interrupt event has not occurred</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 0</th>
<th>SWIF: Software Interrupt Flag bit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 = Software Interrupt Flag Enable</td>
</tr>
<tr>
<td></td>
<td>0 = Software Interrupt Flag Disable</td>
</tr>
</tbody>
</table>

**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

**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.

**Note 2:** IOCIF is a read-only bit. To clear the interrupt condition, all bits in the IOCxIF registers must be cleared.

**Note 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

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Readable</th>
<th>Writable</th>
<th>Unimplemented</th>
<th>Hardware Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>SMT1PWAIF: SMT1 Pulse-Width Acquisition Interrupt Flag bit</td>
<td>1 = Interrupt has occurred (must be cleared by software)</td>
<td>0 = Interrupt event has not occurred</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>SMT1PRAIF: SMT1 Period Acquisition Interrupt Flag bit</td>
<td>1 = Interrupt has occurred (must be cleared by software)</td>
<td>0 = Interrupt event has not occurred</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>SMT1IF: SMT1 Interrupt Flag bit</td>
<td>1 = Interrupt has occurred (must be cleared by software)</td>
<td>0 = Interrupt event has not occurred</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>C1IF: CMP1 Interrupt Flag bit</td>
<td>1 = Interrupt has occurred (must be cleared by software)</td>
<td>0 = Interrupt event has not occurred</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ADTIF: ADC Threshold Interrupt Flag bit</td>
<td>1 = Interrupt has occurred (must be cleared by software)</td>
<td>0 = Interrupt event has not occurred</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>ADIF: ADC Interrupt Flag bit</td>
<td>1 = Interrupt has occurred (must be cleared by software)</td>
<td>0 = Interrupt event has not occurred</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>ZCDIF: ZCD Interrupt Flag bit</td>
<td>1 = Interrupt has occurred (must be cleared by software)</td>
<td>0 = Interrupt event has not occurred</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>INT0IF: External Interrupt 0 Interrupt Flag bit</td>
<td>1 = Interrupt has occurred (must be cleared by software)</td>
<td>0 = Interrupt event has not occurred</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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

**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.

**Note 2:** The external interrupt GPIO pin is selected by the INTxPPS register.
REGISTER 9-5:  PIR2: PERIPHERAL INTERRUPT REGISTER 2\(^{(1)}\)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Readable</th>
<th>Writable</th>
<th>Unimplemented</th>
<th>Value at POR and BOR/Value at all other Resets</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>I2C1RXIF: I(^2)C1 Receive Interrupt Flag bit(^{(2)})</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>SPI1IF: SPI1 Interrupt Flag bit(^{(3)})</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>SPI1TXIF: SPI1 Transmit Interrupt Flag bit(^{(4)})</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>SPI1RXIF: SPI1 Receive Interrupt Flag bit(^{(4)})</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>DMA1AIF: DMA1 Abort Interrupt Flag bit</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>DMA1ORIF: DMA1 Overrun Interrupt Flag bit</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>DMA1DCNTIF: DMA1 Destination Count Interrupt Flag bit</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>DMA1SCNTIF: DMA1 Source Count Interrupt Flag bit</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as ‘0’
- **u** = Bit is unchanged
- **x** = Bit is unknown
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- **HS** = Hardware set
- ‘1’ = interrupt has occurred
- ‘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(1)

<table>
<thead>
<tr>
<th>R/W/HS-0/0</th>
<th>R-0/0</th>
<th>R-0/0</th>
<th>R-0/0</th>
<th>R-0/0</th>
<th>R-0/0</th>
<th>R-0/0</th>
<th>R-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMR0IF</td>
<td>U1IF(2)</td>
<td>U1EIF(3)</td>
<td>U1TXIF(4)</td>
<td>U1RXIF(4)</td>
<td>I2C1EIF(5)</td>
<td>I2C1IF(6)</td>
<td>I2C1TXIF(7)</td>
</tr>
</tbody>
</table>

### 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

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>TMR0IF: TMR0 Interrupt Flag bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Interrupt has occurred (must be cleared by software)</td>
</tr>
<tr>
<td>0</td>
<td>Interrupt event has not occurred</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 6</th>
<th>U1IF: UART1 Interrupt Flag bit(2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Interrupt has occurred</td>
</tr>
<tr>
<td>0</td>
<td>Interrupt event has not occurred</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 5</th>
<th>U1EIF: UART1 Framing Error Interrupt Flag bit(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Interrupt has occurred</td>
</tr>
<tr>
<td>0</td>
<td>Interrupt event has not occurred</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 4</th>
<th>U1TXIF: UART1 Transmit Interrupt Flag bit(4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Interrupt has occurred</td>
</tr>
<tr>
<td>0</td>
<td>Interrupt event has not occurred</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 3</th>
<th>U1RXIF: UART1 Receive Interrupt Flag bit(4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Interrupt has occurred</td>
</tr>
<tr>
<td>0</td>
<td>Interrupt event has not occurred</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 2</th>
<th>I2C1EIF: I2C1 Error Interrupt Flag bit(5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Interrupt has occurred</td>
</tr>
<tr>
<td>0</td>
<td>Interrupt event has not occurred</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 1</th>
<th>I2C1IF: I2C1 Interrupt Flag bit(6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Interrupt has occurred</td>
</tr>
<tr>
<td>0</td>
<td>Interrupt event has not occurred</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 0</th>
<th>I2C1TXIF: I2C1 Transmit Interrupt Flag bit(7)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Interrupt has occurred</td>
</tr>
<tr>
<td>0</td>
<td>Interrupt event has not occurred</td>
</tr>
</tbody>
</table>

### 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.
- **UxIF** is a read-only bit. To clear the interrupt condition, all bits in the UxUIR register must be cleared.
- **UxEIF** is a read-only bit. To clear the interrupt condition, all bits in the UxERRIR register must be cleared.
- **UxTXIF** and **UxRXIF** are read-only bits and cannot be set/cleared by the software.
- **I2CxEIF** is a read-only bit. To clear the interrupt condition, all bits in the I2CxERR register must be cleared.
- **I2CxIF** is a read-only bit. To clear the interrupt condition, all bits in the I2CxPIR register must be cleared.
- **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 (1)**

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLC1IF</td>
<td>CWG1IF</td>
<td>NCO1IF</td>
<td>U-0</td>
<td>CCP1IF</td>
<td>TMR2IF</td>
<td>TMR1GIF</td>
<td>TMR1IF</td>
</tr>
</tbody>
</table>

### 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**

- **CLC1IF**: CLC1 Interrupt Flag bit
  - **1** = Interrupt has occurred (must be cleared by software)
  - **0** = Interrupt event has not occurred

**bit 6**

- **CWG1IF**: CWG1 Interrupt Flag bit
  - **1** = Interrupt has occurred (must be cleared by software)
  - **0** = Interrupt event has not occurred

**bit 5**

- **NCO1IF**: NCO1 Interrupt Flag bit
  - **1** = Interrupt has occurred (must be cleared by software)
  - **0** = Interrupt event has not occurred

**bit 4**

- **Unimplemented**: Read as ‘0’

**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.
## REGISTER 9-8: PIR5: PERIPHERAL INTERRUPT REGISTER 5(1)

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 6</th>
<th>bit 5</th>
<th>bit 4</th>
<th>bit 3</th>
<th>bit 2</th>
<th>bit 1</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2C2TXIF(2)</td>
<td>I2C2RXIF(2)</td>
<td>DMA2AIF</td>
<td>DMA2ORIF</td>
<td>DMA2DCNTIF</td>
<td>DMA2SCNTIF</td>
<td>C2IF</td>
<td>INT1IF(3)</td>
</tr>
</tbody>
</table>

**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 Descriptions:

- **bit 7 I2C2TXIF:** I²C2 Transmit Interrupt Flag bit(2)
  - 1 = Interrupt has occurred
  - 0 = Interrupt event has not occurred

- **bit 6 I2C2RXIF:** I²C2 Receive Interrupt Flag bit(2)
  - 1 = Interrupt has occurred
  - 0 = Interrupt event has not occurred

- **bit 5 DMA2AIF:** DMA2 Abort Interrupt Flag bit
  - 1 = Interrupt has occurred (must be cleared by software)
  - 0 = Interrupt event has not occurred

- **bit 4 DMA2ORIF:** DMA2 Overrun Interrupt Flag bit
  - 1 = Interrupt has occurred (must be cleared by software)
  - 0 = Interrupt event has not occurred

- **bit 3 DMA2DCNTIF:** DMA2 Destination Count Interrupt Flag bit
  - 1 = Interrupt has occurred (must be cleared by software)
  - 0 = Interrupt event has not occurred

- **bit 2 DMA2SCNTIF:** DMA2 Source Count Interrupt Flag bit
  - 1 = Interrupt has occurred (must be cleared by software)
  - 0 = Interrupt event has not occurred

- **bit 1 C2IF:** C2 Interrupt Flag bit
  - 1 = Interrupt has occurred (must be cleared by software)
  - 0 = Interrupt event has not occurred

- **bit 0 INT1IF:** External Interrupt 1 Interrupt Flag bit(3)
  - 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:** I2C2TXIF and I2C2RXIF are read-only bits. To clear the interrupt condition, the CLRBF bit in I2C2STAT1 register must be set.

**3:** The external interrupt GPIO pin is selected by the INTxPPS register.
REGISTER 9-9:  PIR6: PERIPHERAL INTERRUPT REGISTER 6(1)

<table>
<thead>
<tr>
<th>R/W/HS-0/0</th>
<th>R/W/0</th>
<th>R-0/0</th>
<th>R-0/0</th>
<th>R-0/0</th>
<th>R-0/0</th>
<th>R-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMR3GIF</td>
<td>TMR3IF</td>
<td>U2IF(2)</td>
<td>U2EIF(3)</td>
<td>U2TXIF(4)</td>
<td>U2RXIF(4)</td>
<td>I2C2EIF(5)</td>
</tr>
</tbody>
</table>

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  TMR3GIF: TMR3 Gate Interrupt Flag bit
      1 = Interrupt has occurred (must be cleared by software)
      0 = Interrupt event has not occurred

bit 6  TMR3IF: TMR3 Interrupt Flag bit
      1 = Interrupt has occurred (must be cleared by software)
      0 = Interrupt event has not occurred

bit 5  U2IF: UART2 Interrupt Flag bit(2)
      1 = Interrupt has occurred
      0 = Interrupt event has not occurred

bit 4  U2EIF: UART2 Framing Error Interrupt Flag bit(3)
      1 = Interrupt has occurred
      0 = Interrupt event has not occurred

bit 3  U2TXIF: UART2 Transmit Interrupt Flag bit(4)
      1 = Interrupt has occurred
      0 = Interrupt event has not occurred

bit 2  U2RXIF: UART2 Receive Interrupt Flag bit(4)
      1 = Interrupt has occurred
      0 = Interrupt event has not occurred

bit 1  I2C2EIF: I2C2 Error Interrupt Flag bit(5)
      1 = Interrupt has occurred
      0 = Interrupt event has not occurred

bit 0  I2C2IF: I2C2 Interrupt Flag bit(6)
      1 = Interrupt has occurred
      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.
**REGISTER 9-10: PIR7: PERIPHERAL INTERRUPT REGISTER 7**(1)

<table>
<thead>
<tr>
<th>bit 7-6</th>
<th>bit 5</th>
<th>bit 4</th>
<th>bit 3</th>
<th>bit 2</th>
<th>bit 1</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>U-0</td>
<td>U-0</td>
<td>R/W/HS-0/0</td>
<td>R/W/HS-0/0</td>
<td>U-0</td>
<td>R/W/HS-0/0</td>
<td>R/W/HS-0/0</td>
</tr>
<tr>
<td>—</td>
<td>—</td>
<td>INT2IF(2)</td>
<td>CLC2IF</td>
<td>CWG2IF</td>
<td>—</td>
<td>CCP2IF</td>
</tr>
</tbody>
</table>

**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 7-6 Unimplemented:** Read as ‘0’

**bit 5 INT2IF:** External Interrupt 2 Interrupt Flag bit
- 1 = Interrupt has occurred (must be cleared by software)
- 0 = Interrupt event has not occurred

**bit 4 CLC2IF:** CLC2 Interrupt Flag bit
- 1 = Interrupt has occurred (must be cleared by software)
- 0 = Interrupt event has not occurred

**bit 3 CWG2IF:** CWG2 Interrupt Flag bit
- 1 = Interrupt has occurred (must be cleared by software)
- 0 = Interrupt event has not occurred

**bit 2 Unimplemented:** Read as ‘0’

**bit 1 CCP2IF:** CCP2 Interrupt Flag bit
- 1 = Interrupt has occurred (must be cleared by software)
- 0 = Interrupt event has not occurred

**bit 0 TMR4IF:** TMR4 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 9-11: PIR8: PERIPHERAL INTERRUPT REGISTER 8(1)

<table>
<thead>
<tr>
<th>R/W/HS-0/0</th>
<th>R/W/HS-0/0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMR5GIF</td>
<td>TMR5IF</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

Legend:
R = Readable bit
W = Writable bit
U = Unimplemented bit, read as ‘0’
u = Bit is unchanged
x = Bit is unknown
‘1’ = Bit is set
‘0’ = Bit is cleared
HS = Bit is set in hardware

bit 7
TMR5GIF: TMR5 Gate Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 6
TMR5IF: TMR5 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 5-0
Unimplemented: Read as ‘0’

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 9-12: PIR9: PERIPHERAL INTERRUPT REGISTER 9(1)

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W/HS-0/0</th>
<th>R/W/HS-0/0</th>
<th>R/W/HS-0/0</th>
<th>R/W/HS-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CLC3IF</td>
<td>CWG3IF</td>
<td>CCP3IF</td>
<td>TMR6IF</td>
</tr>
</tbody>
</table>

Legend:
R = Readable bit
W = Writable bit
U = Unimplemented bit, read as ‘0’
u = Bit is unchanged
x = Bit is unknown
‘1’ = Bit is set
‘0’ = Bit is cleared

bit 7-4
Unimplemented: Read as ‘0’

bit 3
CLC3IF: CLC3 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 2
CWG3IF: CWG3 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 1
CCP3IF: CCP3 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 0
TMR6IF: TMR6 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 9-13: PIR10: PERIPHERAL INTERRUPT REGISTER 10\(^{(1)}\)

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W/HS-0/0</th>
<th>R/W/HS-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CLC4IF</td>
<td>CCP4IF</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 0</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Legend:</th>
</tr>
</thead>
<tbody>
<tr>
<td>R = Readable bit</td>
</tr>
<tr>
<td>u = Bit is unchanged</td>
</tr>
<tr>
<td>‘1’ = Bit is set</td>
</tr>
</tbody>
</table>

- **bit 7-2**: Unimplemented: Read as ‘0’
- **bit 1**: CLC4IF: CLC4 Interrupt Flag bit
  - 1 = Interrupt has occurred (must be cleared by software)
  - 0 = Interrupt event has not occurred
- **bit 0**: CCP4IF: CCP4 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 9-14: PIE0: PERIPHERAL INTERRUPT ENABLE REGISTER 0

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>IOCIE</td>
<td>Interrupt-on-Change Enable bit</td>
</tr>
<tr>
<td>6</td>
<td>CRCIE</td>
<td>CRC Interrupt Enable bit</td>
</tr>
<tr>
<td>5</td>
<td>SCANIE</td>
<td>Memory Scanner Interrupt Enable bit</td>
</tr>
<tr>
<td>4</td>
<td>NVMIE</td>
<td>NVM Interrupt Enable bit</td>
</tr>
<tr>
<td>3</td>
<td>CSWIE</td>
<td>Clock Switch Interrupt Enable bit</td>
</tr>
<tr>
<td>2</td>
<td>OSFIE</td>
<td>Oscillator Fail Interrupt Enable bit</td>
</tr>
<tr>
<td>1</td>
<td>HLVDIE</td>
<td>HLVD Interrupt Enable bit</td>
</tr>
<tr>
<td>0</td>
<td>SWIE</td>
<td>Software Interrupt Enable bit</td>
</tr>
</tbody>
</table>

**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

- **IOCIE**: Interrupt-on-Change Enable bit
  - 1 = Enabled
  - 0 = Disabled

- **CRCIE**: CRC Interrupt Enable bit
  - 1 = Enabled
  - 0 = Disabled

- **SCANIE**: Memory Scanner Interrupt Enable bit
  - 1 = Enabled
  - 0 = Disabled

- **NVMIE**: NVM Interrupt Enable bit
  - 1 = Enabled
  - 0 = Disabled

- **CSWIE**: Clock Switch Interrupt Enable bit
  - 1 = Enabled
  - 0 = Disabled

- **OSFIE**: Oscillator Fail Interrupt Enable bit
  - 1 = Enabled
  - 0 = Disabled

- **HLVDIE**: HLVD Interrupt Enable bit
  - 1 = Enabled
  - 0 = Disabled

- **SWIE**: Software Interrupt Enable bit
  - 1 = Enabled
  - 0 = Disabled
**REGISTER 9-15: PIE1: PERIPHERAL INTERRUPT ENABLE REGISTER 1**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit 0</th>
<th>Bit 1</th>
<th>Bit 2</th>
<th>Bit 3</th>
<th>Bit 4</th>
<th>Bit 5</th>
<th>Bit 6</th>
<th>Bit 7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>INT0IE</td>
<td>ZCDIE</td>
<td>ADIE</td>
<td>ADTIE</td>
<td>C1IE</td>
<td>SMT1IE</td>
<td>SMT1PRAIE</td>
<td>SMT1PWAIE</td>
</tr>
<tr>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
</tr>
</tbody>
</table>

**Legend:**
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as ‘0’
- n = Value at POR and BOR/Value at all other Resets
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared

- **SMT1PWAIE:** SMT1 Pulse Width Acquisition Interrupt Enable bit
  - 1 = Enabled
  - 0 = Disabled

- **SMT1PRAIE:** SMT1 Period Acquisition Interrupt Enable bit
  - 1 = Enabled
  - 0 = Disabled

- **SMT1IE:** SMT1 Interrupt Enable bit
  - 1 = Enabled
  - 0 = Disabled

- **C1IE:** C1 Interrupt Enable bit
  - 1 = Enabled
  - 0 = Disabled

- **ADTIE:** ADC Threshold Interrupt Enable bit
  - 1 = Enabled
  - 0 = Disabled

- **ADIE:** ADC Interrupt Enable bit
  - 1 = Enabled
  - 0 = Disabled

- **ZCDIE:** ZCD Interrupt Enable bit
  - 1 = Enabled
  - 0 = Disabled

- **INT0IE:** External Interrupt 0 Enable bit
  - 1 = Enabled
  - 0 = Disabled
### REGISTER 9-16: PIE2: PERIPHERAL INTERRUPT ENABLE REGISTER 2

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>I2C1RXIE: I2C1 Receive Interrupt Enable bit</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>SPI1IE: SPI1 Interrupt Enable bit</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>SPI1TXIE: SPI1 Transmit Interrupt Enable bit</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>SPI1RXIE: SPI1 Receive Interrupt Enable bit</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>DMA1AIE: DMA1 Abort Interrupt Enable bit</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>DMA1ORIE: DMA1 Overrun Interrupt Enable bit</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>DMA1DCNTIE: DMA1 Destination Count Interrupt Enable bit</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>DMA1SCNTIE: DMA1 Source Count Interrupt Enable bit</td>
<td></td>
</tr>
</tbody>
</table>

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

- **1** = Enabled
- **0** = Disabled
REGISTER 9-17:  PIE3: PERIPHERAL INTERRUPT ENABLE REGISTER 3

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMR0IE</td>
<td>U1E</td>
</tr>
</tbody>
</table>

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**: TMR0IE: TMR0 Interrupt Enable bit
  - 1 = Enabled
  - 0 = Disabled

- **bit 6**: U1IE: UART1 Interrupt Enable bit
  - 1 = Enabled
  - 0 = Disabled

- **bit 5**: U1EIE: UART1 Framing Error Interrupt Enable bit
  - 1 = Enabled
  - 0 = Disabled

- **bit 4**: U1TXIE: UART1 Transmit Interrupt Enable bit
  - 1 = Enabled
  - 0 = Disabled

- **bit 3**: U1RXIE: UART1 Receive Interrupt Enable bit
  - 1 = Enabled
  - 0 = Disabled

- **bit 2**: I2C1EIE: I2C1 Error Interrupt Enable bit
  - 1 = Enabled
  - 0 = Disabled

- **bit 1**: I2C1IE: I2C1 Interrupt Enable bit
  - 1 = Enabled
  - 0 = Disabled

- **bit 0**: I2C1TXIE: I2C1 Transmit Interrupt Enable bit
  - 1 = Enabled
  - 0 = Disabled
### REGISTER 9-18: PIE4: PERIPHERAL INTERRUPT ENABLE REGISTER 4

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLC1IE</td>
<td>CWG1IE</td>
<td>NCO1IE</td>
<td>—</td>
<td>CCP1IE</td>
<td>TMR2IE</td>
<td>TMR1GIE</td>
<td>TMR1IE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLC1IE: CLC1 Interrupt Enable bit</td>
<td></td>
</tr>
<tr>
<td>1 = Enabled</td>
<td></td>
</tr>
<tr>
<td>0 = Disabled</td>
<td></td>
</tr>
<tr>
<td>CWG1IE: CWG1 Interrupt Enable bit</td>
<td></td>
</tr>
<tr>
<td>1 = Enabled</td>
<td></td>
</tr>
<tr>
<td>0 = Disabled</td>
<td></td>
</tr>
<tr>
<td>NCO1IE: NCO1 Interrupt Enable bit</td>
<td></td>
</tr>
<tr>
<td>1 = Enabled</td>
<td></td>
</tr>
<tr>
<td>0 = Disabled</td>
<td></td>
</tr>
<tr>
<td>CCP1IE: CCP1 Interrupt Enable bit</td>
<td></td>
</tr>
<tr>
<td>1 = Enabled</td>
<td></td>
</tr>
<tr>
<td>0 = Disabled</td>
<td></td>
</tr>
<tr>
<td>TMR2IE: TMR2 Interrupt Enable bit</td>
<td></td>
</tr>
<tr>
<td>1 = Enabled</td>
<td></td>
</tr>
<tr>
<td>0 = Disabled</td>
<td></td>
</tr>
<tr>
<td>TMR1GIE: TMR1 Gate Interrupt Enable bit</td>
<td></td>
</tr>
<tr>
<td>1 = Enabled</td>
<td></td>
</tr>
<tr>
<td>0 = Disabled</td>
<td></td>
</tr>
<tr>
<td>TMR1IE: TMR1 Interrupt Enable bit</td>
<td></td>
</tr>
<tr>
<td>1 = Enabled</td>
<td></td>
</tr>
<tr>
<td>0 = Disabled</td>
<td></td>
</tr>
</tbody>
</table>

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

- **1** = Enabled
- **0** = Disabled
### REGISTER 9-19: PIE5: PERIPHERAL INTERRUPT ENABLE REGISTER 5

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>I2C2TXIE: I²C2 Transmit Interrupt Enable</td>
<td>Enabled</td>
<td>Disabled</td>
</tr>
<tr>
<td>6</td>
<td>I2C2RXIE: I²C2 Receive Interrupt Enable</td>
<td>Enabled</td>
<td>Disabled</td>
</tr>
<tr>
<td>5</td>
<td>DMA2AIE: DMA2 Abort Interrupt Enable</td>
<td>Enabled</td>
<td>Disabled</td>
</tr>
<tr>
<td>4</td>
<td>DMA2ORIE: DMA2 Overrun Interrupt Enable</td>
<td>Enabled</td>
<td>Disabled</td>
</tr>
<tr>
<td>3</td>
<td>DMA2DCNTIE: DMA2 Destination Count Interrupt Enable</td>
<td>Enabled</td>
<td>Disabled</td>
</tr>
<tr>
<td>2</td>
<td>DMA2SCNTIE: DMA2 Source Count Interrupt Enable</td>
<td>Enabled</td>
<td>Disabled</td>
</tr>
<tr>
<td>1</td>
<td>C2IE: C2 Interrupt Enable</td>
<td>Enabled</td>
<td>Disabled</td>
</tr>
<tr>
<td>0</td>
<td>INT1IE: External Interrupt 1 Enable</td>
<td>Enabled</td>
<td>Disabled</td>
</tr>
</tbody>
</table>

**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: I2C2TXIE: I²C2 Transmit Interrupt Enable
  - 1 = Enabled
  - 0 = Disabled

- Bit 6: I2C2RXIE: I²C2 Receive Interrupt Enable
  - 1 = Enabled
  - 0 = Disabled

- Bit 5: DMA2AIE: DMA2 Abort Interrupt Enable
  - 1 = Enabled
  - 0 = Disabled

- Bit 4: DMA2ORIE: DMA2 Overrun Interrupt Enable
  - 1 = Enabled
  - 0 = Disabled

- Bit 3: DMA2DCNTIE: DMA2 Destination Count Interrupt Enable
  - 1 = Enabled
  - 0 = Disabled

- Bit 2: DMA2SCNTIE: DMA2 Source Count Interrupt Enable
  - 1 = Enabled
  - 0 = Disabled

- Bit 1: C2IE: C2 Interrupt Enable
  - 1 = Enabled
  - 0 = Disabled

- Bit 0: INT1IE: External Interrupt 1 Enable
  - 1 = Enabled
  - 0 = Disabled
## REGISTER 9-20: PIE6: PERIPHERAL INTERRUPT ENABLE REGISTER 6

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value at POR</th>
<th>Value at BOR/Value at all other Resets</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>TMR3GIE: TMR3 Gate Interrupt Enable bit</td>
<td>1</td>
<td>-u/n</td>
</tr>
<tr>
<td>6</td>
<td>TMR3IE: TMR3 Interrupt Enable bit</td>
<td>1</td>
<td>-u/n</td>
</tr>
<tr>
<td>5</td>
<td>U2IE: UART2 Interrupt Enable bit</td>
<td>1</td>
<td>-u/n</td>
</tr>
<tr>
<td>4</td>
<td>U2EIE: UART2 Framing Error Interrupt Enable bit</td>
<td>1</td>
<td>-u/n</td>
</tr>
<tr>
<td>3</td>
<td>U2TXIE: UART2 Transmit Interrupt Enable bit</td>
<td>1</td>
<td>-u/n</td>
</tr>
<tr>
<td>2</td>
<td>U2RXIE: UART2 Receive Interrupt Enable bit</td>
<td>1</td>
<td>-u/n</td>
</tr>
<tr>
<td>1</td>
<td>I2C2EIE: i²C2 Error Interrupt Enable bit</td>
<td>1</td>
<td>-u/n</td>
</tr>
<tr>
<td>0</td>
<td>I2C2IE: i²C2 Interrupt Enable bit</td>
<td>1</td>
<td>-u/n</td>
</tr>
</tbody>
</table>

**Legend:**
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as ‘0’
- **u** = Bit is unchanged
- **x** = Bit is unknown
- **‘1’** = Bit is set
- **‘0’** = Bit is cleared
- **-n/n** = Value at POR and BOR/Value at all other Resets
### REGISTER 9-21: PIE7: PERIPHERAL INTERRUPT ENABLE REGISTER 7

<table>
<thead>
<tr>
<th>bit 7-6</th>
<th>bit 5</th>
<th>bit 4</th>
<th>bit 3</th>
<th>bit 2</th>
<th>bit 1</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>U-0</td>
<td>U-0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>U-0</td>
<td>R/W-0/0</td>
</tr>
<tr>
<td>—</td>
<td>—</td>
<td>INT2IE</td>
<td>CLC2IE</td>
<td>CWG2IE</td>
<td>—</td>
<td>CCP2IE</td>
</tr>
</tbody>
</table>

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

- **Unimplemented**: Read as ‘0’

- **INT2IE**: External Interrupt 2 Enable bit
  - 1 = Enabled
  - 0 = Disabled

- **CLC2IE**: CLC2 Interrupt Enable bit
  - 1 = Enabled
  - 0 = Disabled

- **CWG2IE**: CWG2 Interrupt Enable bit
  - 1 = Enabled
  - 0 = Disabled

- **CCP2IE**: CCP2 Interrupt Enable bit
  - 1 = Enabled
  - 0 = Disabled

- **TMR4IE**: TMR4 Interrupt Enable bit
  - 1 = Enabled
  - 0 = Disabled
### REGISTER 9-22: PIE8: PERIPHERAL INTERRUPT ENABLE REGISTER 8

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMR5GIE</td>
<td>TMR5IE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 0</th>
</tr>
</thead>
</table>
| TMR5GIE: TMR5 Gate Interrupt Enable bit
| 1 = Enabled |
| 0 = Disabled |
| bit 6   | TMR5IE: TMR5 Interrupt Enable bit
| 1 = Enabled |
| 0 = Disabled |

### REGISTER 9-23: PIE9: PERIPHERAL INTERRUPT ENABLE REGISTER 9

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CLC3IE</td>
<td>CWG3IE</td>
<td>CCP3IE</td>
<td>TMR6IE</td>
</tr>
</tbody>
</table>

### Legend:

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

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

- **Unimplemented**: Read as ‘0’
**REGISTER 9-24: PIE10: PERIPHERAL INTERRUPT ENABLE REGISTER 10**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td>Unimplemented: Read as '0'</td>
<td></td>
</tr>
<tr>
<td>bit 1</td>
<td>CLC4IE: CLC4 Interrupt Enable bit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = Interrupt has occurred (must be cleared by software)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = Interrupt event has not occurred</td>
<td></td>
</tr>
<tr>
<td>bit 0</td>
<td>CCP4IE: CCP4 Interrupt Enable bit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = Interrupt has occurred (must be cleared by software)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = Interrupt event has not occurred</td>
<td></td>
</tr>
</tbody>
</table>

**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
## REGISTER 9-25: IPR0: PERIPHERAL INTERRUPT PRIORITY REGISTER 0

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>IOCIP: Interrupt-on-Change Priority bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>High priority</td>
</tr>
<tr>
<td>0</td>
<td>Low priority</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 6</th>
<th>CRCIP: CRC Interrupt Priority bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>High priority</td>
</tr>
<tr>
<td>0</td>
<td>Low priority</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 5</th>
<th>SCANIP: Memory Scanner Interrupt Priority bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>High priority</td>
</tr>
<tr>
<td>0</td>
<td>Low priority</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 4</th>
<th>NVMIP: NVM Interrupt Priority bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>High priority</td>
</tr>
<tr>
<td>0</td>
<td>Low priority</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 3</th>
<th>CSWIP: Clock Switch Interrupt Priority bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>High priority</td>
</tr>
<tr>
<td>0</td>
<td>Low priority</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 2</th>
<th>OSFIP: Oscillator Fail Interrupt Priority bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>High priority</td>
</tr>
<tr>
<td>0</td>
<td>Low priority</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 1</th>
<th>HLVDIP: HLVD Interrupt Priority bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>High priority</td>
</tr>
<tr>
<td>0</td>
<td>Low priority</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 0</th>
<th>SWIP: Software Interrupt Priority bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>High priority</td>
</tr>
<tr>
<td>0</td>
<td>Low priority</td>
</tr>
</tbody>
</table>

**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
## REGISTER 9-26: IPR1: PERIPHERAL INTERRUPT PRIORITY REGISTER 1

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
<th>Value</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>SMT1PWAIP</td>
<td>SMT1 Pulse Width Acquisition Interrupt Priority bit</td>
<td>1</td>
<td>High</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>Low</td>
</tr>
<tr>
<td>6</td>
<td>SMT1PRAIP</td>
<td>SMT1 Period Acquisition Interrupt Priority bit</td>
<td>1</td>
<td>High</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>Low</td>
</tr>
<tr>
<td>5</td>
<td>SMT1IP</td>
<td>SMT1 Interrupt Priority bit</td>
<td>1</td>
<td>High</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>Low</td>
</tr>
<tr>
<td>4</td>
<td>C1IP</td>
<td>C1 Interrupt Priority bit</td>
<td>1</td>
<td>High</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>Low</td>
</tr>
<tr>
<td>3</td>
<td>ADTIP</td>
<td>ADC Threshold Interrupt Priority bit</td>
<td>1</td>
<td>High</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>Low</td>
</tr>
<tr>
<td>2</td>
<td>ADIP</td>
<td>ADC Interrupt Priority bit</td>
<td>1</td>
<td>High</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>Low</td>
</tr>
<tr>
<td>1</td>
<td>ZCDIP</td>
<td>ZCD Interrupt Priority bit</td>
<td>1</td>
<td>High</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>Low</td>
</tr>
<tr>
<td>0</td>
<td>INT0IP</td>
<td>External Interrupt 0 Interrupt Priority bit</td>
<td>1</td>
<td>High</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>Low</td>
</tr>
</tbody>
</table>

**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

- **SMT1PWAIP**: SMT1 Pulse Width Acquisition Interrupt Priority bit
  - 1 = High priority
  - 0 = Low priority
- **SMT1PRAIP**: SMT1 Period Acquisition Interrupt Priority bit
  - 1 = High priority
  - 0 = Low priority
- **SMT1IP**: SMT1 Interrupt Priority bit
  - 1 = High priority
  - 0 = Low priority
- **C1IP**: C1 Interrupt Priority bit
  - 1 = High priority
  - 0 = Low priority
- **ADTIP**: ADC Threshold Interrupt Priority bit
  - 1 = High priority
  - 0 = Low priority
- **ADIP**: ADC Interrupt Priority bit
  - 1 = High priority
  - 0 = Low priority
- **ZCDIP**: ZCD Interrupt Priority bit
  - 1 = High priority
  - 0 = Low priority
- **INT0IP**: External Interrupt 0 Interrupt Priority bit
  - 1 = High priority
  - 0 = Low priority
<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>I2C1RXIP: I2C1 Receive Interrupt Priority bit</td>
<td>1 = High priority 0 = Low priority</td>
</tr>
<tr>
<td>6</td>
<td>SPI1IP: SPI1 Transmit Interrupt Priority bit</td>
<td>1 = High priority 0 = Low priority</td>
</tr>
<tr>
<td>5</td>
<td>SPI1TXIP: I2C1 Transmit Interrupt Priority bit</td>
<td>1 = High priority 0 = Low priority</td>
</tr>
<tr>
<td>4</td>
<td>SPI1RXIP: SPI1 Receive Interrupt Priority bit</td>
<td>1 = High priority 0 = Low priority</td>
</tr>
<tr>
<td>3</td>
<td>DMA1AIP: DMA1 Abort Transmit Interrupt Priority bit</td>
<td>1 = High priority 0 = Low priority</td>
</tr>
<tr>
<td>2</td>
<td>DMA1ORIP: DMA1 Overrun Interrupt Priority bit</td>
<td>1 = High priority 0 = Low priority</td>
</tr>
<tr>
<td>1</td>
<td>DMA1DCNTIP: DMA1 Destination Count Interrupt Priority bit</td>
<td>1 = High priority 0 = Low priority</td>
</tr>
<tr>
<td>0</td>
<td>DMA1SCNTIP: DMA1 Source Count Interrupt Priority bit</td>
<td>1 = High priority 0 = Low priority</td>
</tr>
</tbody>
</table>

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
### REGISTER 9-28: IPR3: PERIPHERAL INTERRUPT PRIORITY REGISTER 3

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Legend</th>
<th>Value at POR/BOEn</th>
<th>Value at all other Resets</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>TMR0IP</td>
<td>R = Readable bit</td>
<td>-n/n</td>
<td>'1' = Bit is set; '0' = Bit is cleared</td>
</tr>
<tr>
<td>6</td>
<td>U1IP</td>
<td>W = Writable bit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>U1EIP</td>
<td>U = Unimplemented bit, read as '0'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>U1TXIP</td>
<td>u = Bit is unchanged</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>U1RXIP</td>
<td>x = Bit is unknown</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>I2C1EIP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>I2C1IP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>I2C1TXIP</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Description:
- **TMR0IP:** TMR0 Interrupt Priority bit
  - 1 = High priority
  - 0 = Low priority
- **U1IP:** UART1 Interrupt Priority bit
  - 1 = High priority
  - 0 = Low priority
- **U1EIP:** UART1 Framing Error Interrupt Priority bit
  - 1 = High priority
  - 0 = Low priority
- **U1TXIP:** UART1 Transmit Interrupt Priority bit
  - 1 = High priority
  - 0 = Low priority
- **U1RXIP:** UART1 Receive Interrupt Priority bit
  - 1 = High priority
  - 0 = Low priority
- **I2C1EIP:** I2C1 Error Interrupt Priority bit
  - 1 = High priority
  - 0 = Low priority
- **I2C1IP:** I2C1 Interrupt Priority bit
  - 1 = High priority
  - 0 = Low priority
- **I2C1TXIP:** I2C1 Transmit Interrupt Priority bit
  - 1 = High priority
  - 0 = Low priority
## REGISTER 9-29: IPR4: PERIPHERAL INTERRUPT PRIORITY REGISTER 4

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 6</th>
<th>bit 5</th>
<th>bit 4</th>
<th>bit 3</th>
<th>bit 2</th>
<th>bit 1</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLC1IP</td>
<td>CWG1IP</td>
<td>NCO1IP</td>
<td>—</td>
<td>CCP1IP</td>
<td>TMR2IP</td>
<td>TMR1GIP</td>
<td>TMR1IP</td>
</tr>
</tbody>
</table>

### 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
- **CLC1IP**: CLC1 Interrupt Priority bit
  - 1 = High priority
  - 0 = Low priority

#### bit 6
- **CWG1IP**: CWG1 Interrupt Priority bit
  - 1 = High priority
  - 0 = Low priority

#### bit 5
- **NCO1IP**: NCO1 Interrupt Priority bit
  - 1 = High priority
  - 0 = Low priority

#### bit 4
- **Unimplemented**: Read as ‘0’

#### 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 9-30: IPR5: PERIPHERAL INTERRUPT PRIORITY REGISTER 5

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Readable</th>
<th>Writable</th>
<th>Unimplemented</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>I2C2TXIP: I2C2 Transmit Interrupt Priority bit</td>
<td>1 = High priority</td>
<td>0 = Low priority</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>I2C2RXIP: I2C2 Receive Interrupt Priority bit</td>
<td>1 = High priority</td>
<td>0 = Low priority</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>DMA2AIP: DMA2 Abort Interrupt Priority bit</td>
<td>1 = High priority</td>
<td>0 = Low priority</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>DMA2ORIP: DMA2 Overrun Interrupt Priority bit</td>
<td>1 = High priority</td>
<td>0 = Low priority</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>DMA2DCNTIP: DMA2 Destination Count Interrupt Priority bit</td>
<td>1 = High priority</td>
<td>0 = Low priority</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>DMA2SCNTIP: DMA2 Source Count Interrupt Priority bit</td>
<td>1 = High priority</td>
<td>0 = Low priority</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>C2IP: C2 Interrupt Priority bit</td>
<td>1 = High priority</td>
<td>0 = Low priority</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>INT1IP: External Interrupt 1 Interrupt Priority bit</td>
<td>1 = High priority</td>
<td>0 = Low priority</td>
<td></td>
</tr>
</tbody>
</table>

**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
## REGISTER 9-31: IPR6: PERIPHERAL INTERRUPT PRIORITY REGISTER 6

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMR3GIP</td>
<td>TMR3IP</td>
<td>U2IP</td>
<td>U2EIP</td>
<td>U2TXIP</td>
<td>U2RXIP</td>
<td>I2C2EIP</td>
<td>I2C2IP</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMR3GIP</td>
<td>TMR3 Gate Interrupt Priority bit</td>
<td>1 = High priority, 0 = Low priority</td>
</tr>
<tr>
<td>Bit 6</td>
<td>TMR3IP</td>
<td>TMR3 Interrupt Priority bit</td>
</tr>
<tr>
<td>Bit 5</td>
<td>U2IP</td>
<td>UART2 Interrupt Priority bit</td>
</tr>
<tr>
<td>Bit 4</td>
<td>U2EIP</td>
<td>UART2 Framing Error Interrupt Priority bit</td>
</tr>
<tr>
<td>Bit 3</td>
<td>U2TXIP</td>
<td>UART2 Transmit Interrupt Priority bit</td>
</tr>
<tr>
<td>Bit 2</td>
<td>U2RXIP</td>
<td>UART2 Receive Interrupt Priority bit</td>
</tr>
<tr>
<td>Bit 1</td>
<td>I2C2EIP</td>
<td>I²C2 Error Interrupt Priority bit</td>
</tr>
<tr>
<td>Bit 0</td>
<td>I2C2IP</td>
<td>I²C2 Interrupt Priority bit</td>
</tr>
</tbody>
</table>
### REGISTER 9-32: IPR7: PERIPHERAL INTERRUPT PRIORITY REGISTER 7

<table>
<thead>
<tr>
<th>bit 7-6</th>
<th>bit 5-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPR7</td>
<td></td>
</tr>
<tr>
<td>U-0</td>
<td>U-0</td>
</tr>
<tr>
<td>R/W-1/1</td>
<td>R/W-1/1</td>
</tr>
<tr>
<td>INT2IP</td>
<td>CLC2IP</td>
</tr>
<tr>
<td>CWG2IP</td>
<td>CCP2IP</td>
</tr>
<tr>
<td>TMR4IP</td>
<td></td>
</tr>
</tbody>
</table>

#### 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** INT2IP: External Interrupt 2 Interrupt Priority bit  
  1 = High priority  
  0 = Low priority
- **bit 4** CLC2IP: CLC2 Interrupt Priority bit  
  1 = High priority  
  0 = Low priority
- **bit 3** CWG2IP: CWG2 Interrupt Priority bit  
  1 = High priority  
  0 = Low priority
- **bit 2** Unimplemented: Read as ‘0’
- **bit 1** CCP2IP: CRC Interrupt Priority bit  
  1 = High priority  
  0 = Low priority
- **bit 0** TMR4IP: TMR4 Interrupt Priority bit  
  1 = High priority  
  0 = Low priority

### REGISTER 9-33: IPR8: PERIPHERAL INTERRUPT PRIORITY REGISTER 8

<table>
<thead>
<tr>
<th>bit 7-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPR8</td>
</tr>
<tr>
<td>U-0</td>
</tr>
<tr>
<td>R/W-1/1</td>
</tr>
<tr>
<td>R/W-1/1</td>
</tr>
<tr>
<td>TMR5GIP</td>
</tr>
<tr>
<td>TMR5IP</td>
</tr>
</tbody>
</table>

#### 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** TMR5GIP: TMR5 Gate Interrupt Priority bit  
  1 = High priority  
  0 = Low priority
- **bit 6** TMR5IP: TMR5 Interrupt Priority bit  
  1 = High priority  
  0 = Low priority
- **bit 5-0** Unimplemented: Read as ‘0’
**REGISTER 9-34: IPR9: PERIPHERAL INTERRUPT PRIORITY REGISTER 9**

<table>
<thead>
<tr>
<th></th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CLC3IP</td>
<td>CWG3IP</td>
<td>CCP3IP</td>
<td>TMR6IP</td>
</tr>
</tbody>
</table>

**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** CLC3IP: CLC3 Interrupt Priority bit
  - 1 = High priority
  - 0 = Low priority
- **bit 2** CWG3IP: CWG3 Interrupt Priority bit
  - 1 = High priority
  - 0 = Low priority
- **bit 1** CCP3IP: CCP3 Interrupt Priority bit
  - 1 = High priority
  - 0 = Low priority
- **bit 0** TMR6IP: TMR6 Interrupt Priority bit
  - 1 = High priority
  - 0 = Low priority

**REGISTER 9-35: IPR10: PERIPHERAL INTERRUPT PRIORITY REGISTER 10**

<table>
<thead>
<tr>
<th></th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CLC4IP</td>
<td>CCP4IP</td>
</tr>
</tbody>
</table>

**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** CLC4IP: CLC4 Interrupt Priority bit
  - 1 = High priority
  - 0 = Low priority
- **bit 0** CCP4IP: CCP4 Interrupt Priority bit
  - 1 = High priority
  - 0 = Low priority
REGISTER 9-36: IVTBASEU: INTERRUPT VECTOR TABLE BASE ADDRESS UPPER REGISTER

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>—</td>
<td>—</td>
<td>BASE&lt;20:16&gt;</td>
<td>bit 7</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as ‘0’
- u = Bit is unchanged
- x = Bit is unknown
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- -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 9-37: IVTBASEH: INTERRUPT VECTOR TABLE BASE ADDRESS HIGH REGISTER

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>—</td>
<td>—</td>
<td>BASE&lt;15:8&gt;</td>
<td>bit 7</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

bit 7-0 BASE<15:8>: Interrupt Vector Table Base Address bits

REGISTER 9-38: IVTBASEL: INTERRUPT VECTOR TABLE BASE ADDRESS LOW REGISTER

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-1/1</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>—</td>
<td>—</td>
<td>BASE&lt;7:0&gt;</td>
<td>bit 7</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

bit 7-0 BASE<7:0>: Interrupt Vector Table Base Address bits
### REGISTER 9-39: IVTADU: INTERRUPT VECTOR TABLE ADDRESS UPPER REGISTER

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R-0/0</th>
<th>R-0/0</th>
<th>R-0/0</th>
<th>R-0/0</th>
<th>R-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>AD&lt;20:16&gt;</td>
</tr>
</tbody>
</table>

**Legend:**
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as ’0’
- u = Bit is unchanged
- x = Bit is unknown
- ‘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 9-40: IVTADH: INTERRUPT VECTOR TABLE ADDRESS HIGH REGISTER

<table>
<thead>
<tr>
<th>R-0/0</th>
<th>R-0/0</th>
<th>R-0/0</th>
<th>R-0/0</th>
<th>R-0/0</th>
<th>R-0/0</th>
<th>R-0/0</th>
<th>R-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>AD&lt;15:8&gt;</td>
</tr>
</tbody>
</table>

**Legend:**
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as ’0’
- u = Bit is unchanged
- x = Bit is unknown
- ‘1’ = Bit is set
- ’0’ = Bit is cleared

- bit 7-0: AD<15:8>: Interrupt Vector Table Address bits

### REGISTER 9-41: IVTADL: INTERRUPT VECTOR TABLE ADDRESS LOW REGISTER

<table>
<thead>
<tr>
<th>R-0/0</th>
<th>R-0/0</th>
<th>R-0/0</th>
<th>R-0/0</th>
<th>R-0/0</th>
<th>R-0/0</th>
<th>R-0/0</th>
<th>R-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>AD&lt;7:0&gt;</td>
</tr>
</tbody>
</table>

**Legend:**
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as ’0’
- u = Bit is unchanged
- x = Bit is unknown
- ‘1’ = Bit is set
- ’0’ = Bit is cleared

- bit 7-0: AD<7:0>: Interrupt Vector Table Address bits
REGISTER 9-42:  IVTLOCK: INTERRUPT VECTOR TABLE LOCK REGISTER

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>IVTLOCKED(1,2)</td>
</tr>
</tbody>
</table>

bit 0  IVTLOCKED: IVT Registers Lock bits(1,2)
   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.
       2: If IVT1WAY = 1, the IVTLOCK bit cannot be cleared after it has been set. See Register 5-3.

REGISTER 9-43:  SHADCON: SHADOW CONTROL REGISTER

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SHADLO</td>
</tr>
</tbody>
</table>

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
## TABLE 9-3: SUMMARY OF REGISTERS ASSOCIATED WITH INTERRUPTS

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTCON0</td>
<td>GIE/GIEH</td>
<td>GIE/EIE</td>
<td>IPEN</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>INT2EDG</td>
</tr>
<tr>
<td>INTCON1</td>
<td>STAT&lt;1:0&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>INT2EDG</td>
</tr>
<tr>
<td>PIE0</td>
<td>IIOCIE</td>
<td>CRCIE</td>
<td>SCANIE</td>
<td>NVMIE</td>
<td>CSWIE</td>
<td>OSFIE</td>
<td>—</td>
<td>—</td>
<td>SWIE</td>
</tr>
<tr>
<td>PIE1</td>
<td>SM1TPWAIE</td>
<td>SMT1PRAIE</td>
<td>SM1TIE</td>
<td>C1IE</td>
<td>ADTIE</td>
<td>ADIE</td>
<td>ZCIDE</td>
<td>INT0IE</td>
<td>—</td>
</tr>
<tr>
<td>PIE2</td>
<td>I2C1RXIE</td>
<td>SPI1IE</td>
<td>SPI1TXIE</td>
<td>SPI1RXIE</td>
<td>DMA1IE</td>
<td>DMA1ORIE</td>
<td>DMA1DCNTIE</td>
<td>DMA1SCNTIE</td>
<td>—</td>
</tr>
<tr>
<td>PIE3</td>
<td>TMR0IE</td>
<td>U1IE</td>
<td>U1EIE</td>
<td>U1TXIE</td>
<td>U1RXIE</td>
<td>I2C1IE</td>
<td>I2C1IE</td>
<td>I2C1TXIE</td>
<td>—</td>
</tr>
<tr>
<td>PIE4</td>
<td>CLC1IE</td>
<td>CGW1IE</td>
<td>NCO1IE</td>
<td>—</td>
<td>—</td>
<td>CCP1IE</td>
<td>TMR2IE</td>
<td>TMR1GIE</td>
<td>TMR1IE</td>
</tr>
<tr>
<td>PIE5</td>
<td>I2C2TXIE</td>
<td>I2C2RXIE</td>
<td>DMA2AIE</td>
<td>DMA2ORIE</td>
<td>DMA2DCNTIE</td>
<td>DMA2SCNTIE</td>
<td>C2IE</td>
<td>INT1IE</td>
<td>—</td>
</tr>
<tr>
<td>PIE6</td>
<td>TMR3GIE</td>
<td>TMR3IE</td>
<td>U2IE</td>
<td>U2EIE</td>
<td>U2TXIE</td>
<td>U2RXIE</td>
<td>I2C2IE</td>
<td>I2C2IE</td>
<td>—</td>
</tr>
<tr>
<td>PIE7</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>INT2IE</td>
</tr>
<tr>
<td>PIE8</td>
<td>TMR5GIE</td>
<td>TMR5IE</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>PIE9</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CLC3IE</td>
</tr>
<tr>
<td>PIE10</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CCP2IE</td>
</tr>
<tr>
<td>PIR0</td>
<td>IOCIF</td>
<td>CRCIF</td>
<td>SCANIF</td>
<td>NVMIF</td>
<td>CSWIF</td>
<td>OSFIF</td>
<td>—</td>
<td>—</td>
<td>HLVDIF</td>
</tr>
<tr>
<td>PIR1</td>
<td>SM1TPWAF</td>
<td>SMT1PRAF</td>
<td>SM1TIF</td>
<td>C1IF</td>
<td>ADTIF</td>
<td>ADIF</td>
<td>ZCIF</td>
<td>INT0IF</td>
<td>—</td>
</tr>
<tr>
<td>PIR2</td>
<td>I2C1RXIF</td>
<td>SPI1IF</td>
<td>SPI1TXIF</td>
<td>SPI1RXIF</td>
<td>DMA1IF</td>
<td>DMA1ORIF</td>
<td>DMA1DCNTIF</td>
<td>DMA1SCNTIF</td>
<td>—</td>
</tr>
<tr>
<td>PIR3</td>
<td>TMR0IF</td>
<td>U1IF</td>
<td>U1EIE</td>
<td>U1TXIF</td>
<td>U1RXIF</td>
<td>I2C1IE</td>
<td>I2C1IE</td>
<td>I2C1TXIF</td>
<td>—</td>
</tr>
<tr>
<td>PIR4</td>
<td>CLC1IF</td>
<td>CGW1IF</td>
<td>NCO1IF</td>
<td>—</td>
<td>—</td>
<td>CCP1IF</td>
<td>TMR2IF</td>
<td>TMR1GIF</td>
<td>TMR1IF</td>
</tr>
<tr>
<td>PIR5</td>
<td>I2C2TXF</td>
<td>I2C2RXF</td>
<td>DMA2AIF</td>
<td>DMA2ORIF</td>
<td>DMA2DCNTIF</td>
<td>DMA2SCNTIF</td>
<td>C2IF</td>
<td>INT1IF</td>
<td>—</td>
</tr>
<tr>
<td>PIR6</td>
<td>TMR3GIF</td>
<td>TMR3IF</td>
<td>U2IF</td>
<td>U2EIF</td>
<td>U2TXIF</td>
<td>U2RXIF</td>
<td>I2C2IE</td>
<td>I2C2IE</td>
<td>—</td>
</tr>
<tr>
<td>PIR7</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>INT2IF</td>
</tr>
<tr>
<td>PIR8</td>
<td>TMR5GIF</td>
<td>TMR5IF</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CLC3IF</td>
</tr>
<tr>
<td>PIR9</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CCP3IF</td>
</tr>
<tr>
<td>PIR10</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CCP4IF</td>
</tr>
<tr>
<td>IPR0</td>
<td>IOCIP</td>
<td>CRCIP</td>
<td>SCANIP</td>
<td>NVMIP</td>
<td>CSWIP</td>
<td>OSFIP</td>
<td>—</td>
<td>—</td>
<td>HLVDIP</td>
</tr>
<tr>
<td>IPR1</td>
<td>SM1TPWAIP</td>
<td>SMT1PRAIP</td>
<td>SM1TIP</td>
<td>C1IP</td>
<td>ADTIP</td>
<td>ADIP</td>
<td>ZCIDF</td>
<td>INT0IP</td>
<td>—</td>
</tr>
<tr>
<td>IPR2</td>
<td>I2C1RXIP</td>
<td>SPI1IP</td>
<td>SPI1TXIP</td>
<td>SPI1RXIP</td>
<td>DMA1IP</td>
<td>DMA1ORIP</td>
<td>DMA1DCNTIP</td>
<td>DMA1SCNTIP</td>
<td>—</td>
</tr>
<tr>
<td>IPR3</td>
<td>TMR0IP</td>
<td>U1IP</td>
<td>U1EIP</td>
<td>U1TXIP</td>
<td>U1RXIP</td>
<td>I2C1IE</td>
<td>I2C1IE</td>
<td>I2C1TXIP</td>
<td>—</td>
</tr>
<tr>
<td>IPR4</td>
<td>CLC1IP</td>
<td>CGW1IP</td>
<td>NCO1IP</td>
<td>—</td>
<td>—</td>
<td>CCP1IP</td>
<td>TMR2IP</td>
<td>TMR1GIP</td>
<td>TMR1IP</td>
</tr>
<tr>
<td>IPR5</td>
<td>I2C2TXIP</td>
<td>I2C2RXIP</td>
<td>DMA2AIP</td>
<td>DMA2ORIP</td>
<td>DMA2DCNTIP</td>
<td>DMA2SCNTIP</td>
<td>C2IP</td>
<td>INT1IP</td>
<td>—</td>
</tr>
<tr>
<td>IPR6</td>
<td>TMR3GIP</td>
<td>TMR3IP</td>
<td>U2IP</td>
<td>U2EIP</td>
<td>U2TXIP</td>
<td>U2RXIP</td>
<td>I2C2IE</td>
<td>I2C2IP</td>
<td>—</td>
</tr>
<tr>
<td>IPR7</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>INT2IF</td>
</tr>
<tr>
<td>IPR8</td>
<td>TMR5GIP</td>
<td>TMR5IP</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CLC3IP</td>
</tr>
<tr>
<td>IPR9</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CCP3IP</td>
</tr>
<tr>
<td>IPR10</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CCP4IP</td>
</tr>
<tr>
<td>IVTBASEU</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>BASE&lt;20:16&gt;</td>
</tr>
<tr>
<td>IVTBASEH</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>BASE&lt;15:8&gt;</td>
</tr>
<tr>
<td>IVTBASEL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>BASE&lt;7:0&gt;</td>
</tr>
<tr>
<td>IVTADU</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>AD&lt;20:16&gt;</td>
</tr>
<tr>
<td>IVTADH</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>AD&lt;15:8&gt;</td>
</tr>
<tr>
<td>IVTADL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>AD&lt;7:0&gt;</td>
</tr>
<tr>
<td>IVTLOCK</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>IVTLOCKED</td>
</tr>
</tbody>
</table>

**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 saves power by reducing CPU execution and program memory (PFM) access, without affecting peripheral operation.

10.1.1 DOZE OPERATION

When the Doze Enable bit is set (DOZEN = 1), the CPU executes one instruction cycle out of every N cycles as defined by the DOZE<2:0> bits of the CPUDOZE register. Fosc and Fosc/4 clock sources are unaffected in Doze mode and peripherals can continue using these sources.

10.1.2 INTERRUPTS DURING DOZE

When an interrupt occurs during Doze, the system behavior can be configured using the Recover-On-Interrupt bit (ROI) and the Doze-On-Exit bit (DOE). Refer to Table 10-2 for details about system behavior in all the cases for a transition from Main > ISR > Main. For PIC18(L)F26/27/45/46/47/55/56/57 devices, the transition from Main > ISR > Main always happens in Normal operation, regardless of the state of the DOZEN or DOE bits.

TABLE 10-1: SYSTEM BEHAVIOR FOR INTERRUPT DURING DOZE

<table>
<thead>
<tr>
<th>DOZEN</th>
<th>ROI</th>
<th>Code Flow</th>
<th>Main</th>
<th>ISR (1)</th>
<th>Return to Main</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Normal operation</td>
<td>Normal operation and DOE = DOZEN (in hardware) DOZEN = 0 (unchanged)</td>
<td>If DOE = 1 when return from interrupt; Doze operation and DOZEN = 1 (in hardware)</td>
<td>If DOE = 0 when return from interrupt; Normal operation and DOZEN = 0 (in hardware)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Normal operation</td>
<td>Normal operation and DOE = DOZEN (in hardware) DOZEN = 1 (in hardware)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Doze operation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Doze operation</td>
<td>Normal operation and DOE = DOZEN (in hardware) DOZEN = 0 (in hardware)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note 1: User software can change the DOE bit in ISR.

For example, if ROI = 1 and DOZE<2:0> = 001, the instruction cycle ratio is 1:4. The CPU and memory operate for one instruction cycle and stay idle for the next three instruction cycles. The Doze operation is illustrated in Figure 10-1.
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 CPUUDOZE 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 PD bit of the STATUS register is cleared (Register 4-2)
3. The 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 37.0 “5-Bit Digital-to-Analog Converter (DAC) Module” and Section 34.0 “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 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 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
  - TO bit of the STATUS register will not be set
  - 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
  - TO bit of the STATUS register will be set
  - 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 PD bit. If the PD bit is set, the SLEEP instruction was executed as a NOP.
10.2.3 LOW-POWER SLEEP MODE

The PIC18F26/27/45/46/47/55/56/57K42 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 PIC18F26/27/45/46/47/55/56/57K42 devices allow 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.
10.2.4.1 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.

Note: If CLKOUTEN is enabled (CLKOUTEN = 0, Configuration Word 1H), the output will continue operating while in idle.
## 10.4 Register Definitions: Voltage Regulator Control

**REGISTER 10-1: VREGCON: VOLTAGE REGULATOR CONTROL REGISTER\(^{(1)}\)\)**

<table>
<thead>
<tr>
<th>bit 7-2</th>
<th>bit 1</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>U-0 U-0 U-0 U-0 U-0 R/W-0/0</td>
<td>Reserved</td>
<td>VREGPM</td>
</tr>
</tbody>
</table>

### 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

- **Unimplemented**: Read as ‘0’
- **VREGPM**: Voltage Regulator Power Mode Selection bit
  - **1** = Low-Power Sleep mode enabled in Sleep\(^{(2)}\)
    - Draws lowest current in Sleep, slower wake-up
  - **0** = Normal Power mode enabled in Sleep\(^{(2)}\)
    - Draws higher current in Sleep, faster wake-up

### Note
- **1**: Not present in LF parts.
- **2**: See Section 44.0 “Electrical Specifications”.
**REGISTER 10-2: CPUDOZE: DOZE AND IDLE REGISTER**

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W/HC/HS-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDLEN</td>
<td>DOZEN</td>
<td>ROI</td>
<td>DOE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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; HS = Bit is set by hardware

**bit 7**  
**IDLEN:** Idle Enable bit  
1 = A SLEEP instruction places the device into Idle mode  
0 = A SLEEP instruction places the device into Sleep mode

**bit 6**  
**DOZEN:** Doze Enable bit\(^{(1)}\)  
1 = Places the device into Doze mode  
0 = Places the device into Normal mode

**bit 5**  
**ROI:** Recover-On-Interrupt bit\(^{(1)}\)  
1 = Entering the Interrupt Service Routine (ISR) makes DOZEN = 0  
0 = Entering the Interrupt Service Routine (ISR) does not change DOZEN

**bit 4**  
**DOE:** Doze-On-Exit bit\(^{(1)}\)  
1 = Exiting the Interrupt Service Routine (ISR) makes DOZEN = 1  
0 = Exiting the Interrupt Service Routine (ISR) 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:** Refer Table 10-1 for more details.

**TABLE 10-2: SUMMARY OF REGISTERS ASSOCIATED WITH POWER-DOWN MODE**

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>VREGCON(^{(1)})</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>VREGPM</td>
<td>Reserved</td>
<td>176</td>
</tr>
<tr>
<td>CPUDOZE</td>
<td>IDLEN</td>
<td>DOZEN</td>
<td>ROI</td>
<td>DOE</td>
<td>—</td>
<td>DOZE&lt;2:0&gt;</td>
<td></td>
<td></td>
<td>177</td>
</tr>
</tbody>
</table>

**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 \texttt{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 \texttt{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

 CLRWDT → Window Sizes → WINDOW

 Reserved
 Reserved
 Reserved
 Reserved
 Reserved
 Reserved
 SOsc
 MFINTOSC 31.25 kHz
 LFINTOSC
 CS

 Reserved
 Reserved
 Reserved
 Reserved
 Reserved
 Reserved
 MFINTOSC 31.25 kHz
 LFINTOSC
 CS

 18-bit Prescale Counter

 R

 5-bit WDT Counter

 Overflow Latch → WDT Time-out

 WDTE<1:0> = 01 → SEN

 WDTE<1:0> = 11

 WDTE<1:0> = 10 → Sleep

 Window Sizes

 Comparator

 Window Closed

 WWDT Armed

 WDT Window Violation

 WINDOW

 Reset
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 44.0 “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.

11.3 Time-out Period

If the WDTCPS<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 WDTCPS<4:0> Configuration bits, then the timer period will be based on the WDTCPS<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 WDTCS<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 WDTCWS<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 TO and 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>
<thead>
<tr>
<th>Conditions</th>
<th>WWDT</th>
</tr>
</thead>
<tbody>
<tr>
<td>WDTE&lt;1:0&gt; = 00</td>
<td>Cleared</td>
</tr>
<tr>
<td>WDTE&lt;1:0&gt; = 01 and SEN = 0</td>
<td>Cleared</td>
</tr>
<tr>
<td>WDTE&lt;1:0&gt; = 10 and enter Sleep</td>
<td>Cleared until the end of OST</td>
</tr>
<tr>
<td>CLRWDT Command</td>
<td>Unaffected</td>
</tr>
<tr>
<td>Oscillator Fail Detected</td>
<td>Unaffected</td>
</tr>
<tr>
<td>Exit Sleep + System Clock = SOSC, EXTRC, INTOSC, EXTCLK</td>
<td>Unaffected</td>
</tr>
<tr>
<td>Exit Sleep + System Clock = XT, HS, LP</td>
<td>Unaffected</td>
</tr>
<tr>
<td>Change INTOSC divider (IRCF bits)</td>
<td>Unaffected</td>
</tr>
</tbody>
</table>

FIGURE 11-2: WINDOW PERIOD AND DELAY

![Diagram of Window Period and Delay](image-url)
11.7 Register Definitions: Windowed Watchdog Timer Control

**REGISTER 11-1: WDTCON0: WATCHDOG TIMER CONTROL REGISTER 0**

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W/0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>—</td>
<td>—</td>
<td>PS&lt;4:0&gt;</td>
<td>SEN</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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(1)

<table>
<thead>
<tr>
<th>PS&lt;4:0&gt;</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>11111</td>
<td>Reserved. Results in minimum interval (1:32)</td>
</tr>
<tr>
<td>10011</td>
<td>Reserved. Results in minimum interval (1:32)</td>
</tr>
<tr>
<td>10010</td>
<td>1:8388608 (2^23) (Interval 256s nominal)</td>
</tr>
<tr>
<td>10001</td>
<td>1:4194304 (2^22) (Interval 128s nominal)</td>
</tr>
<tr>
<td>10000</td>
<td>1:2097152 (2^21) (Interval 64s nominal)</td>
</tr>
<tr>
<td>01111</td>
<td>1:1048576 (2^20) (Interval 32s nominal)</td>
</tr>
<tr>
<td>01110</td>
<td>1:524288 (2^19) (Interval 16s nominal)</td>
</tr>
<tr>
<td>01101</td>
<td>1:262144 (2^18) (Interval 8s nominal)</td>
</tr>
<tr>
<td>01100</td>
<td>1:131072 (2^17) (Interval 4s nominal)</td>
</tr>
<tr>
<td>01011</td>
<td>1:65536 (Interval 2s nominal) (Reset value)</td>
</tr>
<tr>
<td>01010</td>
<td>1:32768 (Interval 1s nominal)</td>
</tr>
<tr>
<td>01001</td>
<td>1:16384 (Interval 512 ms nominal)</td>
</tr>
<tr>
<td>01000</td>
<td>1:8192 (Interval 256 ms nominal)</td>
</tr>
<tr>
<td>00111</td>
<td>1:4096 (Interval 128 ms nominal)</td>
</tr>
<tr>
<td>00110</td>
<td>1:2048 (Interval 64 ms nominal)</td>
</tr>
<tr>
<td>00101</td>
<td>1:1024 (Interval 32 ms nominal)</td>
</tr>
<tr>
<td>00100</td>
<td>1:512 (Interval 16 ms nominal)</td>
</tr>
<tr>
<td>00011</td>
<td>1:256 (Interval 8 ms nominal)</td>
</tr>
<tr>
<td>00010</td>
<td>1:128 (Interval 4 ms nominal)</td>
</tr>
<tr>
<td>00001</td>
<td>1:64 (Interval 2 ms nominal)</td>
</tr>
<tr>
<td>00000</td>
<td>1:32 (Interval 1 ms nominal)</td>
</tr>
</tbody>
</table>

**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 WDTCP = <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 WDTCP<4:0> in CONFIG3L.
3: When WDTCP = <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**

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 6-4</th>
<th>bit 3</th>
<th>bit 2-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS&lt;2:0&gt;</td>
<td>WINDOW&lt;2:0&gt;</td>
<td>Unimplemented</td>
<td>Unimplemented</td>
</tr>
</tbody>
</table>

**Legend:**
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as ‘0’
- **u** = Bit is unchanged
- **x** = Bit is unknown
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- **q** = Value depends on condition

**Legend:**
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as ‘0’
- **u** = Bit is unchanged
- **x** = Bit is unknown
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- **q** = Value depends on condition

**Legend:**
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as ‘0’
- **u** = Bit is unchanged
- **x** = Bit is unknown
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- **q** = Value depends on condition

**Legend:**
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as ‘0’
- **u** = Bit is unchanged
- **x** = Bit is unknown
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- **q** = Value depends on condition

**Legend:**
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as ‘0’
- **u** = Bit is unchanged
- **x** = Bit is unknown
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- **q** = Value depends on condition

<table>
<thead>
<tr>
<th>WINDOW&lt;2:0&gt;</th>
<th>Window delay</th>
<th>Window opening</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Percent of time</td>
<td>Percent of time</td>
</tr>
<tr>
<td>111</td>
<td>N/A</td>
<td>100</td>
</tr>
<tr>
<td>110</td>
<td>12.5</td>
<td>87.5</td>
</tr>
<tr>
<td>101</td>
<td>25</td>
<td>75</td>
</tr>
<tr>
<td>100</td>
<td>37.5</td>
<td>62.5</td>
</tr>
<tr>
<td>011</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>010</td>
<td>62.5</td>
<td>37.5</td>
</tr>
<tr>
<td>001</td>
<td>75</td>
<td>25</td>
</tr>
<tr>
<td>000</td>
<td>87.5</td>
<td>12.5</td>
</tr>
</tbody>
</table>

**Note 1:**
- If WDTCCS <2:0> in CONFIG3H = 111, the Reset value of CS<2:0> is 000.

**Note 2:**
- The Reset value of WINDOW<2:0> is determined by the value of WDTCWS<2:0> in the CONFIG3H register.

**Note 3:**
- If WDTCCS<2:0> in CONFIG3H ≠ 111, these bits are read-only.

**Note 4:**
- If WDTCWS<2:0> in CONFIG3H ≠ 111, these bits are read-only.
### REGISTER 11-3: WDTPSL: WWDT PRESCALE SELECT LOW BYTE REGISTER (READ-ONLY)

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 6</th>
<th>bit 5</th>
<th>bit 4</th>
<th>bit 3</th>
<th>bit 2</th>
<th>bit 1</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-0/0</td>
<td>R-0/0</td>
<td>R-0/0</td>
<td>R-0/0</td>
<td>R-0/0</td>
<td>R-0/0</td>
<td>R-0/0</td>
<td>R-0/0</td>
</tr>
</tbody>
</table>

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

**bit 7-0**  
**PSCNT<7:0>:** Prescale Select Low Byte bits\(^{(1)}\)

**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)

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 6</th>
<th>bit 5</th>
<th>bit 4</th>
<th>bit 3</th>
<th>bit 2</th>
<th>bit 1</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-0/0</td>
<td>R-0/0</td>
<td>R-0/0</td>
<td>R-0/0</td>
<td>R-0/0</td>
<td>R-0/0</td>
<td>R-0/0</td>
<td>R-0/0</td>
</tr>
</tbody>
</table>

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

**bit 7-0**  
**PSCNT<15:8>:** Prescale Select High Byte bits\(^{(1)}\)

**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: WDTMR: WDT TIMER REGISTER (READ-ONLY)

<table>
<thead>
<tr>
<th>bit 7-3</th>
<th>WDT TMR&lt;4:0&gt;</th>
<th>STATE</th>
<th>PSCNT&lt;17:16&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7-3</td>
<td>WDT TMR&lt;4:0&gt;</td>
<td>STATE</td>
<td>PSCNT&lt;17:16&gt;</td>
</tr>
</tbody>
</table>

**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**

**WDTTMR<4:0>:** Watchdog Window Value bits

<table>
<thead>
<tr>
<th>WINDOW</th>
<th>WDT Window State</th>
<th>Open Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Closed</td>
<td>Open</td>
</tr>
<tr>
<td>111</td>
<td>00000-11111</td>
<td>100</td>
</tr>
<tr>
<td>110</td>
<td>00000-00011</td>
<td>00100-11111</td>
</tr>
<tr>
<td>101</td>
<td>00000-00111</td>
<td>01000-11111</td>
</tr>
<tr>
<td>100</td>
<td>00000-01011</td>
<td>01100-11111</td>
</tr>
<tr>
<td>011</td>
<td>00000-01111</td>
<td>10000-11111</td>
</tr>
<tr>
<td>010</td>
<td>00000-10011</td>
<td>10100-11111</td>
</tr>
<tr>
<td>001</td>
<td>00000-10111</td>
<td>11000-11111</td>
</tr>
<tr>
<td>000</td>
<td>00000-11011</td>
<td>11100-11111</td>
</tr>
</tbody>
</table>

**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

**Note 1:** The 18-bit WDT prescale value, PSCNT<17:0> includes the WDTPL, 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>WDTCON0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>182</td>
</tr>
<tr>
<td>WDTCON1</td>
<td>—</td>
<td>—</td>
<td>CS&lt;2:0&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>183</td>
</tr>
<tr>
<td>WDTPSL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>184</td>
</tr>
<tr>
<td>WDTPSH</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>184</td>
</tr>
<tr>
<td>WDTTMR</td>
<td>—</td>
<td>—</td>
<td>WDTTMR&lt;4:0&gt;</td>
<td>—</td>
<td>STATE</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>185</td>
</tr>
</tbody>
</table>

**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.

**TABLE 12-1: PERFORMANCE COMPARISON FOR VARIOUS MULTIPLY OPERATIONS**

<table>
<thead>
<tr>
<th>Routine</th>
<th>Multiply Method</th>
<th>Program Memory (Words)</th>
<th>Cycles (Max)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8x8 unsigned</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Without hardware multiply</td>
<td>13</td>
<td>69</td>
<td>4.3 μs</td>
</tr>
<tr>
<td></td>
<td>Hardware multiply</td>
<td>1</td>
<td>1</td>
<td>62.5 ns</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>69 μs</td>
</tr>
<tr>
<td></td>
<td>8x8 signed</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Without hardware multiply</td>
<td>33</td>
<td>91</td>
<td>5.7 μs</td>
</tr>
<tr>
<td></td>
<td>Hardware multiply</td>
<td>6</td>
<td>6</td>
<td>375 ns</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>91 μs</td>
</tr>
<tr>
<td></td>
<td>16x16 unsigned</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Without hardware multiply</td>
<td>21</td>
<td>242</td>
<td>15.1 μs</td>
</tr>
<tr>
<td></td>
<td>Hardware multiply</td>
<td>28</td>
<td>28</td>
<td>1.8 μs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>28 μs</td>
</tr>
<tr>
<td></td>
<td>16x16 signed</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Without hardware multiply</td>
<td>52</td>
<td>254</td>
<td>15.9 μs</td>
</tr>
<tr>
<td></td>
<td>Hardware multiply</td>
<td>35</td>
<td>40</td>
<td>2.5 μs</td>
</tr>
</tbody>
</table>
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**

\[
\text{RES3:RES0} = \text{ARG1H:ARG1L} \cdot \text{ARG2H:ARG2L} \\
= (\text{ARG1H} \cdot \text{ARG2H} \cdot 2^{16}) + \\
(\text{ARG1H} \cdot \text{ARG2L} \cdot 2^8) + \\
(\text{ARG1L} \cdot \text{ARG2H} \cdot 2^8) + \\
(\text{ARG1L} \cdot \text{ARG2L})
\]

**EXAMPLE 12-3: 16 x 16 UNSIGNED MULTIPLY ROUTINE**

```assembly
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
ADDWF RES2, F ;
CLRF WREG ;
ADDWF 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**

\[
\text{RES3:RES0} = \text{ARG1H:ARG1L} \cdot \text{ARG2H:ARG2L} \\
= (\text{ARG1H} \cdot \text{ARG2H} \cdot 2^{16}) + \\
(\text{ARG1H} \cdot \text{ARG2L} \cdot 2^8) + \\
(\text{ARG1L} \cdot \text{ARG2H} \cdot 2^8) + \\
(\text{ARG1L} \cdot \text{ARG2L}) + \\
(-1 \cdot \text{ARG2H}<7> \cdot \text{ARG1H:ARG1L} \cdot 2^{16}) + \\
(-1 \cdot \text{ARG1H}<7> \cdot \text{ARG2H:ARG2L} \cdot 2^8)
\]

**EXAMPLE 12-4: 16 x 16 SIGNED MULTIPLY ROUTINE**

```assembly
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
ADDWF RES2, F ;
CLRF WREG ;
ADDWF RES3, F ;
; BTFS ARG2H, 7 ; ARG2H:ARG2L neg?
BRA SIGN_ARG1 ; no, check ARG1
MOVF ARG1L, W ;
SUBWF RES2 ;
MOVF ARG1H, W ;
SUBWFB RES3
; SIGN_ARG1
BTFS ARG1H, 7 ; ARG1H:ARG1L neg?
BRA CONT_CODE ; no, done
MOVF ARG2L, W ;
SUBWF RES2 ;
MOVF ARG2H, W ;
SUBWFB RES3
; CONT_CODE
```

© 2017 Microchip Technology Inc.
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 (CP and 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.

### TABLE 13-1: NVM ORGANIZATION AND ACCESS INFORMATION

<table>
<thead>
<tr>
<th>Memory</th>
<th>PC&lt;20:0&gt;</th>
<th>ICSP™ Addr&lt;21:0&gt;</th>
<th>TBLPTR&lt;21:0&gt;</th>
<th>NVMADDR&lt;9:0&gt;</th>
<th>Execution</th>
<th>User Access</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CPU</td>
<td>REG</td>
</tr>
<tr>
<td>Program Flash Memory (PFM)</td>
<td>00 0000h</td>
<td>•••</td>
<td>01 FFFFh</td>
<td></td>
<td>Read</td>
<td>10</td>
</tr>
<tr>
<td>User IDs (2)</td>
<td>20 0000h</td>
<td>•••</td>
<td>20 000Fh</td>
<td></td>
<td>No Access</td>
<td>x 1</td>
</tr>
<tr>
<td>Reserved</td>
<td>20 0010h</td>
<td></td>
<td></td>
<td></td>
<td>No Access</td>
<td>_</td>
</tr>
<tr>
<td>Configuration</td>
<td>30 0000h</td>
<td>•••</td>
<td>30 009h</td>
<td></td>
<td>No Access</td>
<td>x 1</td>
</tr>
<tr>
<td>User Data Memory (Data EEPROM)</td>
<td>31 0000h</td>
<td>•••</td>
<td>31 03FFh</td>
<td></td>
<td>No Access</td>
<td>00</td>
</tr>
<tr>
<td>Reserved</td>
<td>31 0400h</td>
<td></td>
<td></td>
<td></td>
<td>No Access</td>
<td>_</td>
</tr>
<tr>
<td>Device Information Area (DIA)</td>
<td>3F 0000h</td>
<td>•••</td>
<td>3F 003Fh</td>
<td></td>
<td>No Access</td>
<td>x 1</td>
</tr>
<tr>
<td>Reserved</td>
<td>3F 0040h</td>
<td></td>
<td></td>
<td></td>
<td>No Access</td>
<td>_</td>
</tr>
<tr>
<td>Device Configuration Information (DCI)</td>
<td>3F FF00h</td>
<td>•••</td>
<td>3F FF09h</td>
<td></td>
<td>No Access</td>
<td>x 1</td>
</tr>
<tr>
<td>Reserved</td>
<td>3F FF0Ah</td>
<td></td>
<td></td>
<td></td>
<td>No Access</td>
<td>_</td>
</tr>
<tr>
<td>Revision ID/Device ID</td>
<td>3F FFFCh</td>
<td>•••</td>
<td>3F FFFFc</td>
<td></td>
<td>No Access</td>
<td>x 1</td>
</tr>
</tbody>
</table>

**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 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.
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.

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 13-1: Table Read Operation](image1)

**Note 1:** Table Pointer register points to a byte in program memory.

**FIGURE 13-2: TABLE WRITE OPERATION**

![Figure 13-2: Table Write Operation](image2)

**Note 1:** During table writes the Table Pointer does not point directly to program memory. The LSBs of TBLPRTL actually point to an address within the write block holding registers. The MSBs of the Table Pointer determine where the write block will eventually be written. The process for writing the holding registers to the program memory array is discussed in Section 13.1.6 "Writing to Program Flash Memory".
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 22nd 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 3, 4, or 5 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 3, 4, or 5 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 13-2: TABLE POINTER OPERATIONS WITH TBLRD AND TBLWT INSTRUCTIONS

<table>
<thead>
<tr>
<th>Example</th>
<th>Operation on Table Pointer</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBLRD*</td>
<td>TBLPTR is not modified</td>
</tr>
<tr>
<td>TBLRD*+</td>
<td>TBLPTR is incremented after the read/write</td>
</tr>
<tr>
<td>TBLRD*-</td>
<td>TBLPTR is decremented after the read/write</td>
</tr>
<tr>
<td>TBLRD+*</td>
<td>TBLPTR is incremented before the read/write</td>
</tr>
<tr>
<td>TBLWT*</td>
<td></td>
</tr>
<tr>
<td>TBLWT*+</td>
<td></td>
</tr>
<tr>
<td>TBLWT*-</td>
<td></td>
</tr>
</tbody>
</table>

FIGURE 13-3: TABLE POINTER BOUNDARIES BASED ON OPERATION

TABLE ERASE/WRITE
TBLPTR<21:n+1>(1)

TABLE WRITE
TBLPTR<n:0>(1)

TABLE READ – TBLPTR<21:0>

Note 1: Refer to Table 5-4 for the row size values.
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
    MOVF TABLAT, W ; get data
    MOVWF WORD_EVEN
    TBLRD*+ ; read into TABLAT and increment
    MOVF TABLAT, W ; get data
    MOVF WORD_ODD
```
FIGURE 13-5: PROGRAM FLASH MEMORY READ FLOWCHART

Start Read Operation

Select PFM (NVMREG<1:0> = 0x10)

Select Word Address (TBLPTR registers)

Initiate Read operation (TBLRD)

Data read now in TABLAT

End Read Operation
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 NVMCON2
- 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.

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 set which 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
    MOVWF TBLPTRU ; load TBLPTR with the base
    MOVWF TBLPTRH
    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
    REQUIRED

    MOVWF NVMCON2 ; write 55h
    MOVWF NVMCON2 ; write AAh

    BSF NVMCON1, WR ; start erase (CPU stalls)
    BSF INTCON0, GIE ; re-enable interrupts
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.
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.
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

```assembly
; number of bytes in erase block
MOVLW D'64';
MOVF COUNTER;
MOVLW BUFFER_ADDR_HIGH;
MOVF FSR0H;
MOVLW BUFFER_ADDR_LOW;
MOVF FSR0L;
MOVLW CODE_ADDR_UPPER;
MOVWF TBLPTRU;
MOVLW CODE_ADDR_HIGH;
MOVWF TBLPTRH;
MOVLW CODE_ADDR_LOW;
MOVWF TBLPTRL;

READ_BLOCK
TBLRD*+;
MOVF TABLAT, W;
MOVWF POSTINC0;
DECFSZ COUNTER;
BRA READ_BLOCK;

MODIFY_WORD
MOVLW BUFFER_ADDR_HIGH;
MOVWF FSR0H;
MOVLW BUFFER_ADDR_LOW;
MOVWF FSR0L;
MOVLW NEW_DATA_LOW;
MOVWF POSTINC0;
MOVLW NEW_DATA_HIGH;
MOVWF INDF0;

ERASE_BLOCK
MOVLW CODE_ADDR_UPPER;
MOVWF TBLPTRU;
MOVLW CODE_ADDR_HIGH;
MOVWF TBLPTRH;
MOVLW CODE_ADDR_LOW;
MOVWF TBLPTRL;
BCF NVMCON1, REG0;
BSF NVMCON1, REG1;
BSF NVMCON1, WREN;
BSF NVMCON1, FREE;
BCF INTCON0, GIE;
MOVLW 55h;
MOVWF NVMCON2;
MOVLW AAh;
MOVWF NVMCON2;
BSF NVMCON1, WR;
BSF INTCON0, GIE;
TBLRD*+;
MOVLW BUFFER_ADDR_HIGH;
MOVWF FSR0H;
MOVLW BUFFER_ADDR_LOW;
MOVWF FSR0L;

WRITE_BUFFER_BACK
MOVLW BlockSize;
MOVWF COUNTER;
MOVLW D'64' / BlockSize;
MOVWF COUNTER2;
```

**Required Sequence**

```assembly
MOVLW 55h;
MOVF NVMCON2;
MOVLW AAh;
MOVWF NVMCON2;
BSF NVMCON1, WR;
BSF INTCON0, GIE;
TBLRD*+;
```
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
DECFSZ COUNTER       ; to internal TBLWT holding register.
BRA WRITE_WORD_TO_HREGS ; loop until holding registers are full

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
MOVWF NVMCON2          ; write 55h
MOVWF NVMCON2          ; write 0AAh
BSF NVMCON1, WR        ; start program (CPU stall)
DCFSZ COUNTER2         ; repeat for remaining write blocks
BRA WRITE_BYTE_TO_HREGS ; re-enable interrupts
BCF NVMCON1, WREN      ; disable write to memory

Required Sequence
FIGURE 13-9: PROGRAM FLASH MEMORY (PFM) WRITE FLOWCHART

Start
Write Operation

Determine number of words to be written into PFM. The number of words cannot exceed the number of words per row (word_cnt)

Select access to PFM locations using NVMREG<1:0> bits

Select Row Address TBLPTR

Select Write Operation (FREE = 0)

Load Write Latches Only

Enable Write/Erase Operation (WREN = 1)

Load the value to write TABLAT

Update the word counter (word_cnt--)

Write Latches to PFM

Last word to write ?

Yes

Disable Interrupts (GIE = 0)

Unlock Sequence(1)

No delay when writing to PFM Latches

Disable Interrupts (GIE = 0)

Unlock Sequence(1)

Re-enable Interrupts (GIE = 1)

Re-enable Interrupts (GIE = 1)

Increment Address TBLPTR++

Re-enable Interrupts (GIE = 1)

CPU stalls while Write operation completes (2 ms typical)

Re-enable Interrupts (GIE = 1)

Disable Write/Erase Operation (WREN = 0)

End Write Operation

Disable Interrupts (GIE = 0)

Unlock Sequence(1)
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 IDs, 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 13-3: DIA, DCI, USER ID, DEV/REV ID AND CONFIGURATION WORD ACCESS (REG<1:0> = x1)

<table>
<thead>
<tr>
<th>Address</th>
<th>Function</th>
<th>Read Access</th>
<th>Write Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>20 0000h-20 000Fh</td>
<td>User IDs</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>30 0000h-30 0009h</td>
<td>Configuration Words</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>3F 0000h-3F 003Fh</td>
<td>DIA</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>3F FF00h-3F FF09h</td>
<td>DCI</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>3F FFFCh-3F FFFFh</td>
<td>Revision ID/Device ID</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
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.

Five SFRs are used to read and write to the data EEPROM as well as the program memory. They are:

- NVMCON1
- NVMCON2
- NVMDAT
- NVMADRHL
- NVMADRHR(1)

13.3.1 NVMADRHL AND NVMADRHR REGISTERS

The NVMADRHL:NVMADRHR 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.

Note 1: NVMADRHR register is not implemented on PIC18(L)F45/55K42.
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.

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.

FIGURE 13-11: DATA EEPROM READ FLOWCHART
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

```assembly
; Data Memory Address to read
CLRF NVMCON1 ; Setup Data EEPROM Access
MOVF EE_ADDRL, W ;
MOVF NVMADRL ; Setup Address
BSF NVMCON1, RD ; Issue EE Read
MOVF NVMDAT, W ; W = EE_DATA
```

EXAMPLE 13-6: DATA EEPROM WRITE

```assembly
; Data Memory Address to write
CLRF NVMCON1 ; Setup Data EEPROM Access
MOVF EE_ADDRL, W ;
MOVF NVMADRL ; Setup Address
; Data Memory Value to write
MOVF EE_DATA, W ;
MOVF NVMDAT ;
; Enable writes
BSF NVMCON1, WREN ;
; Disable interrupts
BCF INTCON0, GIE ;
; Required unlock sequence
MOVLW 55h ;
MOVF NVMCON2 ;
MOVLW AAh ;
MOVF 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 (TPWR).

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
   MOVWF NVMCON2 ; Write 55h
   MOVLF 0AAh ;
   MOVWF NVMCON2 ; Write 0AAh
   BSF NVMCON1, WR ; Set WR bit to begin write
   BTFSC 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**

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>U-0</th>
<th>R/S/HC-0/0</th>
<th>R/W/HS-x/q</th>
<th>R/W-0/0</th>
<th>R/S/HC-0/0</th>
<th>R/S/HC-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>REG&lt;1:0&gt;</td>
<td>___</td>
<td>FREE</td>
<td>WRERR</td>
<td>WREN</td>
<td>WR</td>
<td>RD</td>
<td></td>
</tr>
</tbody>
</table>

**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
- **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**
- **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**
- **1** = A write operation was interrupted by a Reset (hardware set), or WR was written to 0b1 when an invalid address is accessed (Table 9-1, Table 13-1) or WR was written to 0b1 when REG<1:0> and address do not point to the same region or WR was written to 0b1 when a write-protected address is accessed (Table 9-2).
- **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**
- **1** = Initiates an erase/program cycle at the corresponding Data EEPROM Memory location
- **0** = NVM program/erase operation is complete and inactive

**bit 0**

**RD: Read Control bit**
- **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:**
- This can only be used with PFM.
- This bit is set when WR = 1 and clears when the internal programming timer expires or the write is completed successfully.
- Bit must be cleared by the user; hardware will not clear this bit.
- Bit may be written to ‘1’ by the user in order to implement test sequences.
- This bit can only be set by following the unlock sequence of **Section 13.1.4 “NVM Unlock Sequence”**.
- Operations are self-timed and the WR bit is cleared by hardware when complete.
- Once a write operation is initiated, setting this bit to zero will have no effect.
- 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

<table>
<thead>
<tr>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

bit 7-0 ADR<7:0>: EEPROM Read Address bits

REGISTER 13-4: NVMADRH: DATA EEPROM MEMORY ADDRESS HIGH(1)

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

bit 7-2 Unimplemented: Read as ‘0’
bit 1-0 ADR<9:8>: EEPROM Read Address bits

Note 1: The NVMADRH register is not implemented on PIC18(L)F45/55K42.
REGISTER 13-5: NVMDAT: DATA EEPROM MEMORY DATA

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAT&lt;7:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 13-4: SUMMARY OF REGISTERS ASSOCIATED WITH NONVOLATILE MEMORY CONTROL

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>NVCON1</td>
<td>REG&lt;1:0&gt;</td>
<td>—</td>
<td>FREE</td>
<td>WRERR</td>
<td>WREN</td>
<td>WR</td>
<td>RD</td>
<td></td>
<td>210</td>
</tr>
<tr>
<td>NVCON2</td>
<td>Unlock Pattern</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>211</td>
</tr>
<tr>
<td>NVMADRL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>NVMAD&lt;7:0&gt;</td>
<td>—</td>
<td>211</td>
</tr>
<tr>
<td>NVMADRH</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>NVMAD&lt;9:8&gt;</td>
<td>211</td>
</tr>
<tr>
<td>NVMDAT</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>NVMAD&lt;7:0&gt;</td>
<td>—</td>
<td>212</td>
</tr>
</tbody>
</table>

Legend:  
— = unimplemented, read as ‘0’. Shaded bits are not used during EEPROM access.

*Page provides register information.

Note 1: The NVMADRH register is not implemented on PIC18(L)F45/55K42.
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 CRACC<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

```
CRC-16-ANSI
x^16 + x^15 + x^2 + 1 (17 bits)
Standard 16-bit representation = 0x8005
CRCXORH = 0b10000000
CRCXORL = 0b0000010- (1)

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’.
```
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

Linear Feedback Shift Register for CRC-16-ANSI
x^16 + x^15 + x^2 + 1

Augmentation Mode ON

Augmentation Mode OFF

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 theCRCSHIFT 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.
14.5 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 CRXOR 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 CRCACC registers.
3. Program the CRXORH/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 CRCACC 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 CRCACC 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.

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.

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>Bit Name Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRC</td>
<td>CRC</td>
</tr>
</tbody>
</table>

REGISTER 14-1: CRCCON0: CRC CONTROL REGISTER 0

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R-0</th>
<th>R/W-0/0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN</td>
<td>GO</td>
<td>BUSY</td>
<td>ACCM</td>
<td>—</td>
<td>—</td>
<td>SHIFTM</td>
<td>FULL</td>
</tr>
<tr>
<td>bit 7</td>
<td>bit 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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**: CRC Enable bit
1 = CRC module is enabled
0 = CRC is disabled

bit 6  **GO**: CRC Go bit
1 = Start CRC serial shifter
0 = CRC serial shifter turned off

bit 5  **BUSY**: CRC Busy bit
1 = Shifting in progress or pending
0 = All valid bits in shifter have been shifted into accumulator

bit 4  **ACCM**: Accumulator Mode bit
1 = Data is concatenated with zeros
0 = Data is not concatenated with zeros

bit 3-2  **Unimplemented**: Read as ‘0’

bit 1  **SHIFTM**: Shift Mode bit
1 = Shift right (LSb)
0 = Shift left (MSb)

bit 0  **FULL**: Data Path Full Indicator bit
1 = CRCDATH/L registers are full
0 = CRCDATH/L registers have shifted their data into the shifter

REGISTER 14-2: CRCCON1: CRC CONTROL REGISTER 1

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLEN&lt;3:0&gt;</td>
<td></td>
<td>PLEN&lt;3:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 7</td>
<td>bit 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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  **DLEN<3:0>**: Data Length bits
Denotes the length of the data word -1 (See Example 14-1)

bit 3-0  **PLEN<3:0>**: Polynomial Length bits
Denotes the length of the polynomial -1 (See Example 14-1)
### REGISTER 14-3: CRCDATH: CRC DATA HIGH BYTE REGISTER

<table>
<thead>
<tr>
<th>R/W-xx</th>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>bit 0</td>
</tr>
</tbody>
</table>

**Legend:**
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as '0'
- **u** = Bit is unchanged
- **x** = Bit is unknown
- **'1'** = Bit is set
- **'0'** = Bit is cleared

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

**bit 7-0**

**DATA<15:8>:** CRC Input/Output Data bits

### REGISTER 14-4: CRCDATL: CRC DATA LOW BYTE REGISTER

<table>
<thead>
<tr>
<th>R/W-xx</th>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>bit 0</td>
</tr>
</tbody>
</table>

**Legend:**
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as '0'
- **u** = Bit is unchanged
- **x** = Bit is unknown
- **'1'** = Bit is set
- **'0'** = Bit is cleared

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

**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

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>bit 0</td>
</tr>
</tbody>
</table>

**Legend:**
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as '0'
- **u** = Bit is unchanged
- **x** = Bit is unknown
- **'1'** = Bit is set
- **'0'** = Bit is cleared

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

**bit 7-0**

**ACC<15:8>:** CRC Accumulator Register bits
REGISTER 14-6:  CRCACCL: CRC ACCUMULATOR LOW BYTE REGISTER

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC&lt;7:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHIFT&lt;15:8&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHIFT&lt;7:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Readable</th>
<th>Writable</th>
<th>Unimplemented</th>
<th>Value at POR &amp; BOR</th>
<th>Value at other Resets</th>
</tr>
</thead>
<tbody>
<tr>
<td>7-0</td>
<td>X&lt;15:8&gt;: XOR of Polynomial Term X^n Enable bits</td>
<td>R/W-x/x</td>
<td>R/W-x/x</td>
<td>R/W-x/x</td>
<td>R/W-x/x</td>
<td>R/W-x/x</td>
</tr>
</tbody>
</table>

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

### Bit 7-0
- **X<15:8>**: XOR of Polynomial Term X^n Enable bits

## REGISTER 14-10: CRCXORL: CRC XOR LOW BYTE REGISTER

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Readable</th>
<th>Writable</th>
<th>Unimplemented</th>
<th>Value at POR &amp; BOR</th>
<th>Value at other Resets</th>
</tr>
</thead>
<tbody>
<tr>
<td>7-0</td>
<td>X&lt;7:1&gt;: XOR of Polynomial Term X^n Enable bits</td>
<td>R/W-x/x</td>
<td>R/W-x/x</td>
<td>R/W-x/x</td>
<td>R/W-x/x</td>
<td>R/W-x/x</td>
</tr>
</tbody>
</table>

### Legend:
- **R**: Readable bit
- **W**: Writable bit
- **U**: Unimplemented bit
- **u**: Bit is unchanged
- **x**: Bit is unknown
- **'1'**: Bit is set
- **'0'**: Bit is cleared
- **-n/n**: Value at POR and BOR/Value at all other Resets

### Bit 7-0
- **X<7:1>**: XOR of Polynomial Term X^n Enable bits

**Unimplemented**: Read as ‘1’
REGISTER 14-11: SCANCON0: SCANNER ACCESS CONTROL REGISTER 0

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 6</th>
<th>bit 5</th>
<th>bit 4-3</th>
<th>bit 2</th>
<th>bit 1</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN</td>
<td>TRIGEN</td>
<td>SGO</td>
<td>—</td>
<td>—</td>
<td>MREG</td>
<td>BURSTMD</td>
</tr>
</tbody>
</table>

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**: Scanner Enable bit
  - 1 = Scanner is enabled
  - 0 = Scanner is disabled

**bit 6**
- **TRIGEN**: Scanner Trigger Enable bit
  - 1 = Scanner trigger is enabled
  - 0 = Scanner trigger is disabled
  
  Refer Table 14-1.

**bit 5**
- **SGO**: Scanner GO bit
  - 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
  - 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

**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(1)

<table>
<thead>
<tr>
<th>TRIGEN</th>
<th>BURSTMD</th>
<th>Scanner Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Memory access is requested when the CRC module is ready to accept data; the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>request is granted if no other higher priority source request is pending.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Memory access is requested when the CRC module is ready to accept data and trigger</td>
</tr>
<tr>
<td></td>
<td></td>
<td>selection is true; the request is granted if no other higher priority source</td>
</tr>
<tr>
<td></td>
<td></td>
<td>request is pending.</td>
</tr>
<tr>
<td>x</td>
<td>1</td>
<td>Memory access is always requested, the request is granted if no other higher</td>
</tr>
<tr>
<td></td>
<td></td>
<td>priority source request is pending.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>—</td>
<td>LADR&lt;21:16&gt;</td>
<td>(1,2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:
R = Readable bit  W = Writable bit  U = Unimplemented bit, read as ‘0’
\(u\) = Bit is unchanged  \(x\) = Bit is unknown  \(\text{-n/n}\) = 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(1,2)
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 \(\text{SGO = 0}\) (SCANCON0 register).

2: While \(\text{SGO = 1}\) (SCANCON0 register), writing to this register is ignored.

REGISTER 14-13: SCANLADRH: SCAN LOW ADDRESS HIGH BYTE REGISTER

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>LADR&lt;15:8&gt;</td>
<td>(1,2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:
R = Readable bit  W = Writable bit  U = Unimplemented bit, read as ‘0’
\(u\) = Bit is unchanged  \(x\) = Bit is unknown  \(\text{-n/n}\) = 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(1,2)
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 \(\text{SGO = 0}\) (SCANCON0 register).

2: While \(\text{SGO = 1}\) (SCANCON0 register), writing to this register is ignored.
# REGISTER 14-14: SCANLADRL: SCAN LOW ADDRESS LOW BYTE REGISTER

<table>
<thead>
<tr>
<th></th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LADR&lt;7:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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\(^{(1,2)}\)  
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).

**Note 2:** While SGO = 1 (SCANCON0 register), writing to this register is ignored.

---

# REGISTER 14-15: SCANHADRU: SCAN HIGH ADDRESS UPPER BYTE REGISTER

<table>
<thead>
<tr>
<th></th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>—</td>
<td>—</td>
<td>HADR&lt;21:16&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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\(^{(1,2)}\)  
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).

**Note 2:** While SGO = 1 (SCANCON0 register), writing to this register is ignored.
## REGISTER 14-16: SCANHADRH: SCAN HIGH ADDRESS HIGH BYTE REGISTER

<table>
<thead>
<tr>
<th>Bit</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>1/1</td>
<td>1/1</td>
<td>1/1</td>
<td>1/1</td>
<td>1/1</td>
<td>1/1</td>
<td>1/1</td>
</tr>
<tr>
<td>HADR&lt;15:8&gt;: Scan End Address bits(^{(1, 2)})</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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

**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).

**Note 2:** While SGO = 1 (SCANCON0 register), writing to this register is ignored.

## REGISTER 14-17: SCANHADRL: SCAN HIGH ADDRESS LOW BYTE REGISTER

<table>
<thead>
<tr>
<th>Bit</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>1/1</td>
<td>1/1</td>
<td>1/1</td>
<td>1/1</td>
<td>1/1</td>
<td>1/1</td>
<td>1/1</td>
</tr>
<tr>
<td>HADR&lt;7:0&gt;: Scan End Address bits(^{(1, 2)})</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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

**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).

**Note 2:** While SGO = 1 (SCANCON0 register), writing to this register is ignored.
## REGISTER 14-18: SCANTRIG: SCAN TRIGGER SELECTION REGISTER

<table>
<thead>
<tr>
<th></th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>bit 7</td>
<td>TSEL&lt;3:0&gt;</td>
<td>bit 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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

**Unimplemented:** Read as ‘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

- 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRCACCH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>219</td>
</tr>
<tr>
<td>CRCACL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>220</td>
</tr>
<tr>
<td>CRCCON0</td>
<td>EN</td>
<td>GO</td>
<td>BUSY</td>
<td>ACCM</td>
<td>—</td>
<td>—</td>
<td>SHIFT</td>
<td>FULL</td>
<td>218</td>
</tr>
<tr>
<td>CRCCON1</td>
<td>DLEN&lt;3:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PLEN&lt;3:0&gt;</td>
<td></td>
<td>218</td>
</tr>
<tr>
<td>CRCDATH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DATA&lt;15:8&gt;</td>
<td></td>
<td></td>
<td>219</td>
</tr>
<tr>
<td>CRCDATL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DATA&lt;7:0&gt;</td>
<td></td>
<td></td>
<td>219</td>
</tr>
<tr>
<td>CRCSHIFTH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SHIFT&lt;15:8&gt;</td>
<td></td>
<td></td>
<td>220</td>
</tr>
<tr>
<td>CRCSHIFTL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SHIFT&lt;7:0&gt;</td>
<td></td>
<td></td>
<td>220</td>
</tr>
<tr>
<td>CRCXORH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X&lt;15:8&gt;</td>
<td></td>
<td></td>
<td>221</td>
</tr>
<tr>
<td>CRCXORL</td>
<td>X&lt;7:1&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>—</td>
<td></td>
<td>221</td>
</tr>
<tr>
<td>SCANCON0</td>
<td>EN</td>
<td>TRIGEN</td>
<td>SGO</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>MREG</td>
<td>BURSTMD</td>
<td>BUSY</td>
</tr>
<tr>
<td>SCANHADRU</td>
<td>—</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>HADR&lt;21:16&gt;</td>
<td></td>
<td>224</td>
</tr>
<tr>
<td>SCANHADRH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>HADR&lt;15:8&gt;</td>
<td></td>
<td></td>
<td>225</td>
</tr>
<tr>
<td>SCANHADR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>HADR&lt;7:0&gt;</td>
<td></td>
<td></td>
<td>225</td>
</tr>
<tr>
<td>SCANLADRU</td>
<td>—</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>LADR&lt;21:16&gt;</td>
<td></td>
<td>223</td>
</tr>
<tr>
<td>SCANLADR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>LADR&lt;15:8&gt;</td>
<td></td>
<td></td>
<td>223</td>
</tr>
<tr>
<td>SCANLADL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>LADR&lt;7:0&gt;</td>
<td></td>
<td></td>
<td>224</td>
</tr>
<tr>
<td>SCANTRIG</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>TSEL&lt;3:0&gt;</td>
<td>226</td>
</tr>
</tbody>
</table>

**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)F26/27/45/46/47/55/56/57K42 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 K42 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

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>
<thead>
<tr>
<th>TABLE 15-1: DMA MEMORY ACCESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read Source</td>
</tr>
<tr>
<td>-----------------</td>
</tr>
<tr>
<td>Program Flash Memory</td>
</tr>
<tr>
<td>Program Flash Memory</td>
</tr>
<tr>
<td>Data EE</td>
</tr>
<tr>
<td>Data EE</td>
</tr>
<tr>
<td>GPR</td>
</tr>
<tr>
<td>SFR</td>
</tr>
<tr>
<td>GPR</td>
</tr>
<tr>
<td>SFR</td>
</tr>
</tbody>
</table>

**Note:** 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
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.

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.

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 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.

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.
Table 15-2 has a few examples of configuring DMA Message sizes.

**TABLE 15-2: EXAMPLE MESSAGE SIZE TABLE**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Example</th>
<th>SCNT</th>
<th>DCNT</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read from single SFR location to RAM</td>
<td>U1RXB</td>
<td>1</td>
<td>N</td>
<td>N equals the number of bytes desired in the destination buffer. N &gt;= 1</td>
</tr>
<tr>
<td>Write to single SFR location from RAM</td>
<td>U1TXB</td>
<td>N</td>
<td>1</td>
<td>N equals the number of bytes desired in the source buffer. N &gt;= 1</td>
</tr>
<tr>
<td>Read from multiple SFR location</td>
<td>ADRES[H:L]</td>
<td>2</td>
<td>2*N</td>
<td>N equals the number of ADC results to be stored in memory. N&gt;= 1</td>
</tr>
<tr>
<td></td>
<td>TMR1[H:L]</td>
<td>2</td>
<td>2*N</td>
<td>N equals the number of TMR1 Acquisition results to be stored in memory. N&gt;= 1</td>
</tr>
<tr>
<td></td>
<td>SMT1CPR[U:H:L]</td>
<td>3</td>
<td>3*N</td>
<td>N equals the number of Capture Pulse Width measurements to be stored in memory. N&gt;= 1</td>
</tr>
<tr>
<td>Write to Multiple SFR registers</td>
<td>PWMDC[H:L]</td>
<td>2*N</td>
<td>2</td>
<td>N equals the number of PWM duty cycle values to be loaded from a memory table. N&gt;= 1</td>
</tr>
<tr>
<td></td>
<td>All ADC registers</td>
<td>N*31</td>
<td>31</td>
<td>Using the DMA to transfer a complete ADC context from RAM to the ADC registers. N&gt;= 1</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Register</th>
<th>Value loaded</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMAxSPTR&lt;21:0&gt;</td>
<td>DMAxSSA&lt;21:0&gt;</td>
</tr>
<tr>
<td>DMAxSCNT&lt;11:0&gt;</td>
<td>DMAxSSZ&lt;11:0&gt;</td>
</tr>
<tr>
<td>DMAxDPTR&lt;15:0&gt;</td>
<td>DMAxDSA&lt;15:0&gt;</td>
</tr>
<tr>
<td>DMAxDCNT&lt;11:0&gt;</td>
<td>DMAxDSZ&lt;11:0&gt;</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Register</th>
<th>Modified Source Counter/Pointer Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMAxSCNT&lt;11:0&gt;</td>
<td>DMAxSCNT = DMAxSCNT -1</td>
</tr>
<tr>
<td>SMODE = 00:</td>
<td>DMAxSPTR = DMAxSPTR</td>
</tr>
<tr>
<td>SMODE = 01:</td>
<td>DMAxSPTR = DMAxSPTR + 1</td>
</tr>
<tr>
<td>SMODE = 10:</td>
<td>DMAxSPTR = DMAxSPTR - 1</td>
</tr>
<tr>
<td>DMAxSCNT&lt;11:0&gt;</td>
<td>DMAxSCNT = DMAxSSZ</td>
</tr>
<tr>
<td>DMAxSPTR</td>
<td>DMAxSSA</td>
</tr>
</tbody>
</table>

### TABLE 15-5: DMA DESTINATION POINTER/COUNTER DURING OPERATION

<table>
<thead>
<tr>
<th>Register</th>
<th>Modified Destination Counter/Pointer Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMAxDCNT&lt;11:0&gt;</td>
<td>DMAxDCNT = DMAxDCNT -1</td>
</tr>
<tr>
<td>DMODE = 00:</td>
<td>DMAxDPTR = DMAxDPTR</td>
</tr>
<tr>
<td>DMODE = 01:</td>
<td>DMAxDPTR = DMAxDPTR + 1</td>
</tr>
<tr>
<td>DMODE = 10:</td>
<td>DMAxDPTR = DMAxDPTR - 1</td>
</tr>
<tr>
<td>DMAxDCNT&lt;11:0&gt;</td>
<td>DMAxDCNT = DMAxDSZ</td>
</tr>
<tr>
<td>DMAxDPTR</td>
<td>DMAxDSA</td>
</tr>
</tbody>
</table>

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.
- **Note 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 in 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
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.

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

1. Configure DMA Module
   - EN = 1

2. Load DMA Source/ Destination Pointers & Counters

3. SIRQEN = 1 & Trigger?
   - Y: DGO = 1
   - N

4. Bubble?
   - Y: Source Read
   - N

5. Destination Write
   - DMAxBUF = &DMAxSPTR
   - KIP = 1
   - &DMAxDPTR = DMABUF
   - KIP = 0

6. DMAxSCNT = 0
   - Y: Reload DMAxSCNT & DMAxSPTR
   - N

7. Update DMAxSA, DMAxSCNT

8. DMAxDCNT = 0
   - Y: Reload DMAxDCNT & DMAxDPTR
   - N

9. Update DMAxSA, DMAxDCNT

10. DGO = 0
    - Y: End Process
    - N
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**

**Note 1:** SR - Source Read  
**2:** DW - Destination Write
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

Note 1: SR - Source Read
2: DW - Destination Write
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**

**Note 1:** SR - Source Read  
**Note 2:** DW - Destination Write
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

**Note 1:** SR - Source Read

**2:** DW - Destination Write
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**

<table>
<thead>
<tr>
<th>Instruction Clock</th>
<th>EN</th>
<th>SIRQEN</th>
<th>Source Hardware Trigger</th>
<th>DGO</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DMAxSPTR</th>
<th>DMAxDPTR</th>
<th>DMAxSCNT</th>
<th>DMAxDCNT</th>
<th>DMAxSTATE</th>
<th>DMAxSCNTIF</th>
<th>DMAxDCNTIF</th>
<th>DMAxORIF</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x100</td>
<td>0x200</td>
<td>2</td>
<td>4</td>
<td>IDLE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x101</td>
<td>0x201</td>
<td>1</td>
<td>3</td>
<td>SR(1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x100</td>
<td>0x202</td>
<td>1</td>
<td>2</td>
<td>DW(2)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x101</td>
<td>0x203</td>
<td>1</td>
<td>2</td>
<td>IDLE</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note 1: SR - Source Read
2: DW - Destination Write
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**

<table>
<thead>
<tr>
<th>Instruction Clock</th>
<th>EN</th>
<th>SIRQEN</th>
<th>AIRQEN</th>
<th>Source Hardware Trigger</th>
<th>Abort Hardware Trigger</th>
<th>DGO</th>
<th>DMAxSPTR</th>
<th>DMAxDPTR</th>
<th>DMAxSCNT</th>
<th>DMAxDCNT</th>
<th>DMAxSCNTIF</th>
<th>DMAxDCNTIF</th>
<th>DMAxAIF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>0x3EEF</td>
<td>0x3EF0</td>
<td>0x3EEF</td>
<td>0x3EF0</td>
<td>0x3EEF</td>
<td>0x3EEF</td>
<td>0x3EEF</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>0x100</td>
<td>0x101</td>
<td>0x109</td>
<td>0x10A</td>
<td>0x100</td>
<td>0x109</td>
<td>0x10A</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>9</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DMA STATE</td>
<td>IDLE</td>
<td>SR(1)</td>
<td>DW(2)</td>
<td>IDLE</td>
<td>SR(1)</td>
<td>DW(2)</td>
<td>IDLE</td>
<td>SR(1)</td>
<td>DW(2)</td>
<td>IDLE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DMAxSCNTIF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DMAxDCNTIF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DMAxAIF</td>
<td>DMAxSSA</td>
<td>0x3EEF</td>
<td>DMAxDA</td>
<td>0x100</td>
<td>DMAxSSZ</td>
<td>0x2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DMAxDSZ</td>
<td>0xA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note 1:** SR - Source Read  
**2:** DW - Destination Write
15.9.7 ABORT TRIGGER, MESSAGE IN PROGRESS

When an abort interrupt request is received in 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>
<thead>
<tr>
<th>Instruction Clock</th>
<th>EN</th>
<th>SIRQEN</th>
<th>AIRQEN</th>
<th>Source Hardware Trigger</th>
<th>Abort Hardware Trigger</th>
<th>DGO</th>
<th>DMAxSPTR</th>
<th>DMAxDPTR</th>
<th>DMAxSCNT</th>
<th>DMAxDCNT</th>
<th>DMA STATE</th>
<th>DMAxCONbits.XIP</th>
<th>DMAxAIF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x3EEF</td>
<td>0x100</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x3EEF</td>
<td>0x101</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x3EEF</td>
<td>0x102</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x3EF0</td>
<td>0x101</td>
<td></td>
<td></td>
<td></td>
<td>9</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x3EF0</td>
<td>0x102</td>
<td></td>
<td></td>
<td></td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

Note 1: SR - Source Read
Note 2: DW - Destination Write
<table>
<thead>
<tr>
<th>Source Module</th>
<th>Source Register(s)</th>
<th>Destination Module</th>
<th>Destination Register(s)</th>
<th>DCHxSIRQ</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal Measurement Timer (SMT)</td>
<td>SMTxCPW[U:H:L]</td>
<td>GPR</td>
<td>GPR[x,y,z]</td>
<td>SMTxPWAIF</td>
<td>Store Captured Pulse-width values</td>
</tr>
<tr>
<td></td>
<td>SMTxCPR[U:H:L]</td>
<td></td>
<td></td>
<td>SMTxPRAIF</td>
<td>Store Captured Period values</td>
</tr>
<tr>
<td>GPR/SFR/Program Flash/Data EEPROM</td>
<td>MEMORY[x,y]</td>
<td>TMR0</td>
<td>TMR0[H:L]</td>
<td>TMR0IF</td>
<td>Use as a Timer0 reload for custom 16-bit value</td>
</tr>
<tr>
<td></td>
<td>GPR/SFR/Program Flash/Data EEPROM</td>
<td>MEMORY[x]</td>
<td>TMR0</td>
<td>PR0</td>
<td>ANY Update TMR0 frequency based on a specific trigger</td>
</tr>
<tr>
<td>TMR1</td>
<td>TMR1[H:L]</td>
<td>GPR</td>
<td>GPR[x,y]</td>
<td>TMR1GIF</td>
<td>Use TMR1 Gate interrupt flag to read data out of TMR1 register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GPR/SFR/Program Flash/Data EEPROM</td>
<td>MEMORY[x]</td>
<td>TMR2</td>
<td>PR2</td>
<td>TMR2IF</td>
<td>Frequency generator with 50% duty cycle look up table</td>
</tr>
<tr>
<td>GPR/SFR/Program Flash/Data EEPROM</td>
<td>MEMORY[x,y,z]</td>
<td>TMR2</td>
<td>PR2</td>
<td>ANY</td>
<td>Load Compare value or PWM values into the CCP</td>
</tr>
<tr>
<td>GPR/SFR/Program Flash/Data EEPROM</td>
<td>MEMORY [x,y,z,u,v,w]</td>
<td>CCPx</td>
<td>CCPx[H:L]</td>
<td>ANY</td>
<td>Update multiple PWM values at the same time e.g. 3-phase motor control</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CCPy</td>
<td>CCPy[H:L]</td>
<td></td>
<td>Frequency Generator look-up table</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CCPz</td>
<td>CCPz[H:L]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GPR/SFR/Program Flash/Data EEPROM</td>
<td>MEMORY[x,y,z]</td>
<td>NCO</td>
<td>NCOxINC[U:H:L]</td>
<td>ANY</td>
<td>Update DAC values</td>
</tr>
<tr>
<td>GPR/SFR/Program Flash/Data EEPROM</td>
<td>MEMORY[x]</td>
<td>DAC</td>
<td>DACxCON0</td>
<td>ANY</td>
<td>Automated Frequency dithering</td>
</tr>
<tr>
<td>GPR/SFR/Program Flash/Data EEPROM</td>
<td>MEMORY[x]</td>
<td>OSCTUNE</td>
<td>OSCTUNE</td>
<td>ANY</td>
<td></td>
</tr>
</tbody>
</table>
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 Bank 64 (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 Bank 64 Registers.
**EXAMPLE 15-1:** SETUP DMA1 TO MOVE DATA FROM PROGRAM FLASH MEMORY TO UART1 TRANSMIT BUFFER USING HARDWARE TRIGGERS

```c
//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**

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>Bit Name Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMA 1</td>
<td>DMA1</td>
</tr>
<tr>
<td>DMA 2</td>
<td>DMA2</td>
</tr>
</tbody>
</table>
### REGISTER 15-1: DMAxCON0: DMAx CONTROL REGISTER 0

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value at POR and BOR</th>
<th>Value at all other Resets</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>EN: DMA Module Enable bit</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>SIRQEN: Start of Transfer Interrupt Request Enable bits</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>DGO: DMA transaction bit</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4-3</td>
<td>Unimplemented: Read as ‘0’</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>AIRQEN: Abort of Transfer Interrupt Request Enable bits</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Unimplemented: Read as ‘0’</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>XIP: Transfer in Progress Status bit</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

#### 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

- **EN**: DMA Module Enable bit
  - 1 = Enables module
  - 0 = Disables module

- **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

- **DGO**: DMA transaction bit
  - 1 = DMA transaction is in progress
  - 0 = DMA transaction is not in progress

- **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

- **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

<table>
<thead>
<tr>
<th>Bit</th>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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 |
| 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 |
| 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 |
| 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 |
| 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 |

**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
### REGISTER 15-3: DMAxBUF: DMAx DATA BUFFER REGISTER

<table>
<thead>
<tr>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUF7</td>
<td>BUF6</td>
<td>BUF5</td>
<td>BUF4</td>
<td>BUF3</td>
<td>BUF2</td>
<td>BUF1</td>
<td>BUF0</td>
</tr>
</tbody>
</table>

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 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

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSA&lt;7:0&gt;</td>
<td>SSA&lt;7:0&gt;</td>
<td>SSA&lt;7:0&gt;</td>
<td>SSA&lt;7:0&gt;</td>
<td>SSA&lt;7:0&gt;</td>
<td>SSA&lt;7:0&gt;</td>
<td>SSA&lt;7:0&gt;</td>
</tr>
</tbody>
</table>

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 SSA<7:0>: Source Start Address bits**

### REGISTER 15-5: DMAxSSAH: DMAx SOURCE START ADDRESS HIGH REGISTER

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSA&lt;15:8&gt;</td>
<td>SSA&lt;15:8&gt;</td>
<td>SSA&lt;15:8&gt;</td>
<td>SSA&lt;15:8&gt;</td>
<td>SSA&lt;15:8&gt;</td>
<td>SSA&lt;15:8&gt;</td>
<td>SSA&lt;15:8&gt;</td>
<td>SSA&lt;15:8&gt;</td>
</tr>
</tbody>
</table>

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 SSA<15:8>: Source Start Address bits**
**REGISTER 15-6:**  DMAxSSAU: DMAx SOURCE START ADDRESS UPPER REGISTER

<table>
<thead>
<tr>
<th></th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td>—</td>
<td>—</td>
<td>SSA&lt;21:16&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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 SSA<21:16>: Source Start Address bits**

**REGISTER 15-7:**  DMAxSPTRL: DMAx SOURCE POINTER LOW REGISTER

<table>
<thead>
<tr>
<th></th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td>—</td>
<td>—</td>
<td>SPTR&lt;7:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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 15-0 SPTR<7:0>: Current Source Address Pointer**

**REGISTER 15-8:**  DMAxSPTRH: DMAx SOURCE POINTER HIGH REGISTER

<table>
<thead>
<tr>
<th></th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td>—</td>
<td>—</td>
<td>SPTR&lt;15:8&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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 5-0 SPTR<15:8>: Current Source Address Pointer**
REG 15-9: DMAxSPTRU: DMAx SOURCE POINTER UPPER REGISTER

<table>
<thead>
<tr>
<th></th>
<th>U-0</th>
<th>U-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7-6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 7-6 Unimplemented: Read as ‘0’
- bit 5-0 SPTR<21:16>: Current Source Address Pointer

REG 15-10: DMAxSSZL: DMAx SOURCE SIZE LOW REGISTER

<table>
<thead>
<tr>
<th></th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
</table>
| bit 7-0 | SSZ<7:0>: Source Message Size bits

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 SSZ<7:0>: Source Message Size bits
## REGISTER 15-12: DMAxSCNTL: DMAx SOURCE COUNT LOW REGISTER

<table>
<thead>
<tr>
<th>bit 7-0</th>
<th>SCNT&lt;7:0&gt;: Current Source Byte Count</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCNT&lt;7:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 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

### 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

### bit 7-0
- SCNT<7:0>: Current Source Byte Count

## REGISTER 15-13: DMAxSCNTH: DMAx SOURCE COUNT HIGH REGISTER

<table>
<thead>
<tr>
<th>bit 7-0</th>
<th>SCNT&lt;11:8&gt;: Current Source Byte Count</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCNT&lt;11:8&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 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

### 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

<table>
<thead>
<tr>
<th>bit 7-0</th>
<th>DSA&lt;7:0&gt;: Destination Start Address bits</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSA&lt;7:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 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

### bit 7-0
- DSA<7:0>: Destination Start Address bits
### REGISTER 15-15: DMAxDSAH: DMAx DESTINATION START ADDRESS HIGH REGISTER

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DSA&lt;15:8&gt;</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSA&lt;15:8&gt;: Destination Start Address bits</td>
<td></td>
</tr>
</tbody>
</table>

### REGISTER 15-16: DMAxDPTRL: DMAx DESTINATION POINTER LOW REGISTER

<table>
<thead>
<tr>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>D PTR&lt;7:0&gt;</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>D PTR&lt;7:0&gt;: Current Destination Address Pointer</td>
<td></td>
</tr>
</tbody>
</table>

### REGISTER 15-17: DMAxDPTRH: DMAx DESTINATION POINTER HIGH REGISTER

<table>
<thead>
<tr>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>D PTR&lt;15:8&gt;</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>D PTR&lt;15:8&gt;: Current Destination Address Pointer</td>
<td></td>
</tr>
</tbody>
</table>
### REGISTER 15-18: DMAxDSZL: DMAx DESTINATION SIZE LOW REGISTER

<table>
<thead>
<tr>
<th></th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td>DSZ&lt;7:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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
- **u** = bit is unchanged

- **DSZ<7:0>**: Destination Message Size bits

### REGISTER 15-19: DMAxDSZH: DMAx DESTINATION SIZE HIGH REGISTER

<table>
<thead>
<tr>
<th></th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DSZ&lt;11:8&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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
- **u** = bit is unchanged

- **DSZ<11:8>**: Destination Message Size bits
- **Unimplemented**: Read as ‘0’

### REGISTER 15-20: DMAxDCNTL: DMAx DESTINATION COUNT LOW REGISTER

<table>
<thead>
<tr>
<th></th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DCNT&lt;7:0&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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
- **u** = bit is unchanged

- **DCNT<7:0>**: Current Destination Byte Count
**REGISTER 15-21: DMAxDCNTH: DMAx DESTINATION COUNT HIGH REGISTER**

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>—</td>
<td>—</td>
<td>DCNT&lt;11:8&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

bit 7

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
- u = bit is unchanged
- -n/n = Value at POR and BOR/Value at all other Resets

bit 7-4 Unimplemented: Read as ‘0’
bit 3-0 DCNT<11:8>: Current Destination Byte Count

**REGISTER 15-22: DMAxSIRQ: DMAx START INTERRUPT REQUEST SOURCE SELECTION REGISTER**

<table>
<thead>
<tr>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>SIRQ&lt;6:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

bit 7

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
- u = bit is unchanged
- -n/n = Value at POR and BOR/Value at all other Resets

bit 7 Unimplemented: Read as ‘0’
bit 6-0 SIRQ<6:0>: DMAx Start Interrupt Request Source Selection bits
Please refer to Table 15-2 for more information.

**REGISTER 15-23: DMAxAIRQ: DMAx ABORT INTERRUPT REQUEST SOURCE SELECTION REGISTER**

<table>
<thead>
<tr>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>AIRQ&lt;6:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

bit 7

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
- u = bit is unchanged
- -n/n = Value at POR and BOR/Value at all other Resets

bit 7 Unimplemented: Read as ‘0’
bit 6-0 AIRQ<6:0>: DMAx Abort Interrupt Request Source Selection bits
Please refer to Table 15-2 for more information.
### TABLE 15-2: DMAxSIRQ AND DMAxAIRQ INTERRUPT SOURCES

<table>
<thead>
<tr>
<th>DMAxSIRQ</th>
<th>Trigger Source</th>
<th>Level Triggered</th>
<th>DMAxAIRQ</th>
<th>Trigger Source</th>
<th>Level Triggered</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Reserved</td>
<td></td>
<td>42</td>
<td>DMA2SCNT</td>
<td>No</td>
</tr>
<tr>
<td>1</td>
<td>LVD</td>
<td>No</td>
<td>43</td>
<td>DMA2DCNT</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>OSF</td>
<td>No</td>
<td>44</td>
<td>DMA2OR</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>CSW</td>
<td>No</td>
<td>45</td>
<td>DMA2A</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>NVM</td>
<td>No</td>
<td>46</td>
<td>I2C2RX</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>SCAN</td>
<td>No</td>
<td>47</td>
<td>I2C2TX</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>CRC</td>
<td>No</td>
<td>48</td>
<td>I2C2</td>
<td>Yes</td>
</tr>
<tr>
<td>7</td>
<td>IOC</td>
<td>Yes</td>
<td>49</td>
<td>I2C2E</td>
<td>Yes</td>
</tr>
<tr>
<td>8</td>
<td>INT0</td>
<td>No</td>
<td>50</td>
<td>U2RX</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>ZCD</td>
<td>No</td>
<td>51</td>
<td>U2TX</td>
<td>Yes</td>
</tr>
<tr>
<td>10</td>
<td>AD</td>
<td>No</td>
<td>52</td>
<td>U2E</td>
<td>Yes</td>
</tr>
<tr>
<td>11</td>
<td>ADT</td>
<td>No</td>
<td>53</td>
<td>U2</td>
<td>No</td>
</tr>
<tr>
<td>12</td>
<td>CMP1</td>
<td>No</td>
<td>54</td>
<td>TMR3</td>
<td>No</td>
</tr>
<tr>
<td>13</td>
<td>SMT1</td>
<td>No</td>
<td>55</td>
<td>TMR3G</td>
<td>No</td>
</tr>
<tr>
<td>14</td>
<td>SMT1PRA</td>
<td>No</td>
<td>56</td>
<td>TMR4</td>
<td>No</td>
</tr>
<tr>
<td>15</td>
<td>SMT1PWA</td>
<td>No</td>
<td>57</td>
<td>CCP2</td>
<td>No</td>
</tr>
<tr>
<td>16</td>
<td>DMA1SCNT</td>
<td>No</td>
<td>58</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>DMA1DCNT</td>
<td>No</td>
<td>59</td>
<td>CWG2</td>
<td>No</td>
</tr>
<tr>
<td>18</td>
<td>DMA1OR</td>
<td>No</td>
<td>60</td>
<td>CLC2</td>
<td>No</td>
</tr>
<tr>
<td>19</td>
<td>DMA1A</td>
<td>No</td>
<td>61</td>
<td>INT2</td>
<td>No</td>
</tr>
<tr>
<td>20</td>
<td>SPI1RX</td>
<td>Yes</td>
<td>62</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>SPI1TX</td>
<td>Yes</td>
<td>63</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>SPI1</td>
<td>Yes</td>
<td>64</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>I2C1RX</td>
<td>Yes</td>
<td>65</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>I2C1TX</td>
<td>Yes</td>
<td>66</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>I2C1</td>
<td>Yes</td>
<td>67</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>I2C1E</td>
<td>Yes</td>
<td>68</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>U1RX</td>
<td>Yes</td>
<td>69</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>U1TX</td>
<td>Yes</td>
<td>70</td>
<td>TMR5</td>
<td>No</td>
</tr>
<tr>
<td>29</td>
<td>U1E</td>
<td>Yes</td>
<td>71</td>
<td>TMR5G</td>
<td>No</td>
</tr>
<tr>
<td>30</td>
<td>U1</td>
<td>No</td>
<td>72</td>
<td>TMR6</td>
<td>No</td>
</tr>
<tr>
<td>31</td>
<td>TMR0</td>
<td>No</td>
<td>73</td>
<td>CCP3</td>
<td>No</td>
</tr>
<tr>
<td>32</td>
<td>TMR1</td>
<td>No</td>
<td>74</td>
<td>CWG3</td>
<td>No</td>
</tr>
<tr>
<td>33</td>
<td>TMR1G</td>
<td>No</td>
<td>75</td>
<td>CLC3</td>
<td>No</td>
</tr>
<tr>
<td>34</td>
<td>TMR2</td>
<td>No</td>
<td>76</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>CCP1</td>
<td>No</td>
<td>77</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>Reserved</td>
<td></td>
<td>78</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>NCO</td>
<td>No</td>
<td>79</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>CWG1</td>
<td>No</td>
<td>80</td>
<td>CCP4</td>
<td>No</td>
</tr>
<tr>
<td>39</td>
<td>CLC1</td>
<td>No</td>
<td>81</td>
<td>CLC4</td>
<td>No</td>
</tr>
<tr>
<td>40</td>
<td>INT1</td>
<td>No</td>
<td>82</td>
<td>–</td>
<td>Reserved</td>
</tr>
<tr>
<td>41</td>
<td>CMP2</td>
<td>No</td>
<td></td>
<td>127</td>
<td></td>
</tr>
</tbody>
</table>

**Note 1:** All trigger sources that are not Level-triggered are Edge-triggered.
### TABLE 15-3: SUMMARY OF REGISTERS ASSOCIATED WITH DMA

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMAxCON0</td>
<td>EN</td>
<td>SIRQEN</td>
<td>DGO</td>
<td></td>
<td></td>
<td>AIRQEN</td>
<td></td>
<td>XIP</td>
<td>248</td>
</tr>
<tr>
<td>DMAxCON1</td>
<td>DMODE&lt;1:0&gt;</td>
<td>DSTP</td>
<td>SMR&lt;1:0&gt;</td>
<td>SMODE&lt;1:0&gt;</td>
<td>SSTP</td>
<td></td>
<td></td>
<td></td>
<td>249</td>
</tr>
<tr>
<td>DMAxBUF</td>
<td>DBUF7</td>
<td>DBUF6</td>
<td>DBUF5</td>
<td>DBUF4</td>
<td>DBUF3</td>
<td>DBUF2</td>
<td>DBUF1</td>
<td>DBUF0</td>
<td>250</td>
</tr>
<tr>
<td>DMAxSSAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>250</td>
</tr>
<tr>
<td>DMAxSSAH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>250</td>
</tr>
<tr>
<td>DMAxSSAU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SSA&lt;1:6&gt;</td>
<td></td>
<td></td>
<td>251</td>
</tr>
<tr>
<td>DMAxSPTRL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SPTR&lt;7:0&gt;</td>
<td></td>
<td></td>
<td>251</td>
</tr>
<tr>
<td>DMAxSPTRH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SPTR&lt;15:8&gt;</td>
<td></td>
<td></td>
<td>251</td>
</tr>
<tr>
<td>DMAxSPTRU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SPTR&lt;1:16&gt;</td>
<td></td>
<td></td>
<td>252</td>
</tr>
<tr>
<td>DMAxSSZL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SSZ&lt;7:0&gt;</td>
<td></td>
<td></td>
<td>252</td>
</tr>
<tr>
<td>DMAxSSZH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SSZ&lt;11:8&gt;</td>
<td></td>
<td></td>
<td>252</td>
</tr>
<tr>
<td>DMAxSCNTL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SCNT&lt;7:0&gt;</td>
<td></td>
<td></td>
<td>253</td>
</tr>
<tr>
<td>DMAxSCNTH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SCNT&lt;11:8&gt;</td>
<td></td>
<td></td>
<td>253</td>
</tr>
<tr>
<td>DMAxDSAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DSA&lt;7:0&gt;</td>
<td></td>
<td></td>
<td>253</td>
</tr>
<tr>
<td>DMAxDSAH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DSA&lt;15:8&gt;</td>
<td></td>
<td></td>
<td>254</td>
</tr>
<tr>
<td>DMAxDPTRL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>D PTR&lt;7:0&gt;</td>
<td></td>
<td></td>
<td>254</td>
</tr>
<tr>
<td>DMAxDPTRH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>D PTR&lt;15:8&gt;</td>
<td></td>
<td></td>
<td>254</td>
</tr>
<tr>
<td>DMAxDSZL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DSZ&lt;7:0&gt;</td>
<td></td>
<td></td>
<td>255</td>
</tr>
<tr>
<td>DMAxDSZH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DSZ&lt;11:8&gt;</td>
<td></td>
<td></td>
<td>255</td>
</tr>
<tr>
<td>DMAxCNTL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DCNT&lt;7:0&gt;</td>
<td></td>
<td></td>
<td>255</td>
</tr>
<tr>
<td>DMAxCNTH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DCNT&lt;11:8&gt;</td>
<td></td>
<td></td>
<td>256</td>
</tr>
<tr>
<td>DMAxSIRQ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SIRQ&lt;6:0&gt;</td>
<td></td>
<td></td>
<td>256</td>
</tr>
<tr>
<td>DMAxAIRQ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>AIRQ&lt;6:0&gt;</td>
<td></td>
<td></td>
<td>256</td>
</tr>
</tbody>
</table>

**Legend:** — = unimplemented location, read as ‘0’. Shaded cells are not used by DMA.
16.0 I/O PORTS

The PIC18(L)F26/27/45/46/47/55/56/57K42 devices have six I/O ports, allocated as shown in Table 16-1.

TABLE 16-1: PORT ALLOCATION TABLE FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES

<table>
<thead>
<tr>
<th>Device</th>
<th>PORTA</th>
<th>PORTB</th>
<th>PORTC</th>
<th>PORTD</th>
<th>PORTE</th>
<th>PORTF</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIC18(L)F26K42</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td>•</td>
</tr>
<tr>
<td>PIC18(L)F27K42</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td>•</td>
</tr>
<tr>
<td>PIC18(L)F45K42</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td>•</td>
</tr>
<tr>
<td>PIC18(L)F46K42</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td>•</td>
</tr>
<tr>
<td>PIC18(L)F47K42</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td>•</td>
</tr>
<tr>
<td>PIC18(L)F55K42</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td>•</td>
</tr>
<tr>
<td>PIC18(L)F56K42</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td>•</td>
</tr>
<tr>
<td>PIC18(L)F57K42</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
</tbody>
</table>

Note 1: Pin RE3 only.
2: Pins RE0, RE1, RE2 and 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 (input level control)
- SLRCONx registers (slew rate control)
- ODCONx registers (open-drain 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 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

16.2 PORTx Registers

In this section, the generic names such as PORTx, LATx, TRISx, etc. can be associated with PORTA, PORTB, and PORTC. The functionality of PORTE is different compared to other ports and is explained in a separate section.
16.2.1 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 PORTA 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**

```assembly
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
MOVF TRISA ;and set RA<2:0> as
             ;outputs
```

16.2.2 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’.

16.2.3 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.

16.2.4 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 I2C.

16.2.5 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.
16.2.6 INPUT THRESHOLD CONTROL

The INLVLx register (Register 16-8) controls the input voltage threshold for each of the available PORTx 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 PORTx register and also the level at which an interrupt-on-change occurs, if that feature is enabled. See Table 44-6 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.

16.2.7 WEAK PULL-UP CONTROL

The WPUx register (Register 16-5) controls the individual weak pull-ups for each port pin.

16.2.8 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. For further details about the IOC module refer to Section 18.0 “Interrupt-on-Change”.

16.2.9 I^2C PAD CONTROL

For the PIC18(L)F26/27/45/46/47/55/56/57K42 devices, the I^2C specific pads are available on RB1, RB2, RC3, RC4, RD0(1) and RD1(1) pins. The I^2C characteristics of each of these pins is controlled by the RxyI2C registers (see Register 16-9). These characteristics include enabling I^2C specific slew rate (over standard GPIO slew rate), selecting internal pull-ups for I^2C pins, and selecting appropriate input threshold as per SMBus specifications.

Note 1: RD0 and RD1 I^2C pads are not available in PIC18(L)F26K42 parts.

Note 2: Any peripheral using the I^2C pins read the I^2C ST inputs when enabled via RxyI2C.

16.3 PORTE Registers

Depending on the device, PORTE is implemented in two different ways.

16.3.1 PORTE ON 40/44/48-PIN DEVICES

For PIC18(L)F45/46/47/55/56/57K42 devices, PORTE is a 4-bit wide port. Three pins (RE0, RE1 and RE2) are individually configurable as inputs or outputs. These pins have Schmitt Trigger input buffers. When selected as an analog input, these pins will read as ‘0’s. The corresponding data direction register is TRISE. Setting a TRISE bit (= 1) will make the corresponding PORTE pin an input (i.e., disable the output driver).

Clearing a TRISE bit (= 0) will make the corresponding PORTE pin an output (i.e., enable the output driver and put the contents of the output latch on the selected pin). TRISE controls the direction of the REx pins, even when they are being used as analog pins. The user must make sure to keep the pins configured as inputs when using them as analog inputs. RE<2:0> bits have other registers associated with them (i.e., ANSELE, WPUE, INLVLE, SLRCONE and ODCONE). The functionality is similar to the other ports. The Data Latch register (LATE) is also memory-mapped. Read-modify-write operations on the LATE register read and write the latched output value for PORTE.

Note: On a Power-on Reset, RE<2:0> are configured as analog inputs.

The fourth pin of PORTE (MCLR/VPP/RE3) is an input-only pin. Its operation is controlled by the MCLRE Configuration bit. When selected as a port pin, (MCLRE = 0), it functions as a digital input-only pin; as such, it does not have TRIS or LAT bits associated with its operation. Otherwise, it functions as the device’s Master Clear input. In either configuration, RE3 also functions as the programming voltage input during programming. RE3 in PORTE register is a read-only bit and will read ‘1’ when MCLRE = 1 (i.e., Master Clear enabled).

Note: On a Power-on Reset, RE3 is enabled as a digital input only if Master Clear functionality is disabled.

Note 1: RD0 and RD1 I^2C pads are not available in PIC18(L)F26K42 parts.

Note 2: Any peripheral using the I^2C pins read the I^2C ST inputs when enabled via RxyI2C.
EXAMPLE 16-2: INITIALIZING PORTE

CLRF PORTE ;Initialize PORTE by
   ; clearing output
   ; data latches
CLRF LATE ;Alternate method
   ; to clear output
   ; data latches
CLRF ANSELE ;Configure analog pins
   ; for digital only
MOVlw 05h ;Value used to
   ; initialize data
   ; direction
MOVWF TRISE ;Set RE<0> as input
   ; RE<1> as output
   ; RE<2> as input

16.3.2 PORTE ON 28-PIN DEVICES

For PIC18(L)F26/27K42 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).

16.3.3 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 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.

16.3.4 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”.

CLRF PORTE ;Initialize PORTE by
   ; clearing output
   ; data latches
CLRF LATE ;Alternate method
   ; to clear output
   ; data latches
CLRF ANSELE ;Configure analog pins
   ; for digital only
MOVlw 05h ;Value used to
   ; initialize data
   ; direction
MOVWF TRISE ;Set RE<0> as input
   ; RE<1> as output
   ; RE<2> as input
## 16.4 Register Definitions: Port Control

### REGISTER 16-1: PORTx: PORTx REGISTER\(^{(1)}\)

<table>
<thead>
<tr>
<th>R/W-x/u</th>
<th>Rx7</th>
<th>Rx6</th>
<th>Rx5</th>
<th>Rx4</th>
<th>Rx3</th>
<th>Rx2</th>
<th>Rx1</th>
<th>Rx0</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### 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

#### 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PORTA</td>
<td>RA7</td>
<td>RA6</td>
<td>RA5</td>
<td>RA4</td>
<td>RA3</td>
<td>RA2</td>
<td>RA1</td>
<td>RA0</td>
</tr>
<tr>
<td>PORTB</td>
<td>RB7(^{(1)})</td>
<td>RB6(^{(1)})</td>
<td>RB5</td>
<td>RB4</td>
<td>RB3</td>
<td>RB2</td>
<td>RB1</td>
<td>RB0</td>
</tr>
<tr>
<td>PORTC</td>
<td>RC7</td>
<td>RC6</td>
<td>RC5</td>
<td>RC4</td>
<td>RC3</td>
<td>RC2</td>
<td>RC1</td>
<td>RC0</td>
</tr>
<tr>
<td>PORTD(^{(3)})</td>
<td>RD7</td>
<td>RD6</td>
<td>RD5</td>
<td>RD4</td>
<td>RD3</td>
<td>RD2</td>
<td>RD1</td>
<td>RD0</td>
</tr>
<tr>
<td>PORTE</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>RE3(^{(2)})</td>
<td>RE2(^{(3)})</td>
<td>RE1(^{(3)})</td>
<td>RE0(^{(3)})</td>
</tr>
<tr>
<td>PORTF(^{(4)})</td>
<td>RF7</td>
<td>RF6</td>
<td>RF5</td>
<td>RF4</td>
<td>RF3</td>
<td>RF2</td>
<td>RF1</td>
<td>RF0</td>
</tr>
</tbody>
</table>

#### Notes:
- Bits RB6 and RB7 read \('1'\) while in Debug mode.
- Bit PORTE3 is read-only, and will read \('1'\) when MCLRE = 1 (Master Clear enabled).
- Unimplemented in PIC18(L)F26/27K42.
REGISTER 16-2: TRISx: TRI-STATE CONTROL REGISTER

<table>
<thead>
<tr>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRISx7</td>
<td>TRISx6</td>
<td>TRISx5</td>
<td>TRISx4</td>
<td>TRISx3</td>
<td>TRISx2</td>
<td>TRISx1</td>
</tr>
</tbody>
</table>

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

Table 16-3: TRIS Registers

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRISA</td>
<td>TRISA7</td>
<td>TRISA6</td>
<td>TRISA5</td>
<td>TRISA4</td>
<td>TRISA3</td>
<td>TRISA2</td>
<td>TRISA1</td>
<td>TRISA0</td>
</tr>
<tr>
<td>TRISB</td>
<td>TRISB7(1)</td>
<td>TRISB6(1)</td>
<td>TRISB5</td>
<td>TRISB4</td>
<td>TRISB3</td>
<td>TRISB2</td>
<td>TRISB1</td>
<td>TRISB0</td>
</tr>
<tr>
<td>TRISC</td>
<td>TRISC7</td>
<td>TRISC6</td>
<td>TRISC5</td>
<td>TRISC4</td>
<td>TRISC3</td>
<td>TRISC2</td>
<td>TRISC1</td>
<td>TRISC0</td>
</tr>
<tr>
<td>TRISD(2)</td>
<td>TRISD7</td>
<td>TRISD6</td>
<td>TRISD5</td>
<td>TRISD4</td>
<td>TRISD3</td>
<td>TRISD2</td>
<td>TRISD1</td>
<td>TRISD0</td>
</tr>
<tr>
<td>TRISE(2)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>TRISE2</td>
<td>TRISE1</td>
</tr>
<tr>
<td>TRISF(3)</td>
<td>TRISF7</td>
<td>TRISF6</td>
<td>TRISF5</td>
<td>TRISF4</td>
<td>TRISF3</td>
<td>TRISF2</td>
<td>TRISF1</td>
<td>TRISF0</td>
</tr>
</tbody>
</table>

Note:
1: Bits RB6 and RB7 read ‘1’ while in Debug mode.
2: Unimplemented in PIC18(L)F26/27K42.
REGISTER 16-3:  LATx: LATx REGISTER(1)

<table>
<thead>
<tr>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
</tr>
</thead>
<tbody>
<tr>
<td>LATx7</td>
<td>LATx6</td>
<td>LATx5</td>
<td>LATx4</td>
<td>LATx3</td>
<td>LATx2</td>
<td>LATx1</td>
<td>LATx0</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>LATA</td>
<td>LATA7</td>
<td>LATA6</td>
<td>LATA5</td>
<td>LATA4</td>
<td>LATA3</td>
<td>LATA2</td>
<td>LATA1</td>
<td>LATA0</td>
</tr>
<tr>
<td>LATB</td>
<td>LATB7</td>
<td>LATB6</td>
<td>LATB5</td>
<td>LATB4</td>
<td>LATB3</td>
<td>LATB2</td>
<td>LATB1</td>
<td>LATB0</td>
</tr>
<tr>
<td>LATC</td>
<td>LATC7</td>
<td>LATC6</td>
<td>LATC5</td>
<td>LATC4</td>
<td>LATC3</td>
<td>LATC2</td>
<td>LATC1</td>
<td>LATC0</td>
</tr>
<tr>
<td>LATD(1)</td>
<td>LATD7</td>
<td>LATD6</td>
<td>LATD5</td>
<td>LATD4</td>
<td>LATD3</td>
<td>LATD2</td>
<td>LATD1</td>
<td>LATD0</td>
</tr>
<tr>
<td>LATE(1)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>LATE2</td>
<td>LATE1</td>
</tr>
<tr>
<td>LATF(2)</td>
<td>LATF7</td>
<td>LATF6</td>
<td>LATF5</td>
<td>LATF4</td>
<td>LATF3</td>
<td>LATF2</td>
<td>LATF1</td>
<td>LATF0</td>
</tr>
</tbody>
</table>

Note 1: Unimplemented in PIC18(L)F26/27K42.
2: Unimplemented in PIC18(L)F26/45/46/47K42.
**REGISTER 16-4: ANSELx: ANALOG SELECT REGISTER**

<table>
<thead>
<tr>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSELx7</td>
<td>ANSELx6</td>
<td>ANSELx5</td>
<td>ANSELx4</td>
<td>ANSELx3</td>
<td>ANSELx2</td>
<td>ANSELx1</td>
<td>ANSELx0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 6</th>
<th>bit 5</th>
<th>bit 4</th>
<th>bit 3</th>
<th>bit 2</th>
<th>bit 1</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSELA</td>
<td>ANSELA7</td>
<td>ANSELA6</td>
<td>ANSELA5</td>
<td>ANSELA4</td>
<td>ANSELA3</td>
<td>ANSELA2</td>
<td>ANSELA1</td>
</tr>
<tr>
<td>ANSELB</td>
<td>ANSELB7</td>
<td>ANSELB6</td>
<td>ANSELB5</td>
<td>ANSELB4</td>
<td>ANSELB3</td>
<td>ANSELB2</td>
<td>ANSELB1</td>
</tr>
<tr>
<td>ANSELC</td>
<td>ANSELC7</td>
<td>ANSELC6</td>
<td>ANSELC5</td>
<td>ANSELC4</td>
<td>ANSELC3</td>
<td>ANSELC2</td>
<td>ANSELC1</td>
</tr>
<tr>
<td>ANSELF<a href="1">@</a></td>
<td>ANSELF7</td>
<td>ANSELF6</td>
<td>ANSELF5</td>
<td>ANSELF4</td>
<td>ANSELF3</td>
<td>ANSELF2</td>
<td>ANSELF1</td>
</tr>
<tr>
<td>ANSELF<a href="2">@</a></td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ANSELF2</td>
<td>ANSELF1</td>
</tr>
</tbody>
</table>

**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**

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSELA</td>
<td>ANSELA7</td>
<td>ANSELA6</td>
<td>ANSELA5</td>
<td>ANSELA4</td>
<td>ANSELA3</td>
<td>ANSELA2</td>
<td>ANSELA1</td>
<td>ANSELA0</td>
</tr>
<tr>
<td>ANSELB</td>
<td>ANSELB7</td>
<td>ANSELB6</td>
<td>ANSELB5</td>
<td>ANSELB4</td>
<td>ANSELB3</td>
<td>ANSELB2</td>
<td>ANSELB1</td>
<td>ANSELB0</td>
</tr>
<tr>
<td>ANSELC</td>
<td>ANSELC7</td>
<td>ANSELC6</td>
<td>ANSELC5</td>
<td>ANSELC4</td>
<td>ANSELC3</td>
<td>ANSELC2</td>
<td>ANSELC1</td>
<td>ANSELC0</td>
</tr>
<tr>
<td>ANSELF<a href="1">@</a></td>
<td>ANSELF7</td>
<td>ANSELF6</td>
<td>ANSELF5</td>
<td>ANSELF4</td>
<td>ANSELF3</td>
<td>ANSELF2</td>
<td>ANSELF1</td>
<td>ANSELF0</td>
</tr>
<tr>
<td>ANSELF<a href="2">@</a></td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ANSELF2</td>
<td>ANSELF1</td>
<td>ANSELF0</td>
</tr>
</tbody>
</table>

**Note:**

1: Unimplemented in PIC18(L)F26/27K42.
2: Unimplemented in PIC18(L)F26/45/46/47K42.
**REGISTER 16-5: WPUx: WEAK PULL-UP REGISTER**

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>WPUx7</td>
<td>WPUx6</td>
<td>WPUx5</td>
<td>WPUx4</td>
<td>WPUx3</td>
<td>WPUx2</td>
<td>WPUx1</td>
<td>WPUx0</td>
</tr>
</tbody>
</table>

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

**TABLE 16-6: WEAK PULL-UP PORT REGISTERS**

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>WPUA</td>
<td>WPUA7</td>
<td>WPUA6</td>
<td>WPUA5</td>
<td>WPUA4</td>
<td>WPUA3</td>
<td>WPUA2</td>
<td>WPUA1</td>
<td>WPUA0</td>
</tr>
<tr>
<td>WPUB</td>
<td>WPUB7</td>
<td>WPUB6</td>
<td>WPUB5</td>
<td>WPUB4</td>
<td>WPUB3</td>
<td>WPUB2</td>
<td>WPUB1</td>
<td>WPUB0</td>
</tr>
<tr>
<td>WPUC</td>
<td>WPUC7</td>
<td>WPUC6</td>
<td>WPUC5</td>
<td>WPUC4</td>
<td>WPUC3</td>
<td>WPUC2</td>
<td>WPUC1</td>
<td>WPUC0</td>
</tr>
<tr>
<td>WPUD²</td>
<td>WPUD7</td>
<td>WPUD6</td>
<td>WPUD5</td>
<td>WPUD4</td>
<td>WPUD3</td>
<td>WPUD2</td>
<td>WPUD1</td>
<td>WPUD0</td>
</tr>
<tr>
<td>WPUE</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>WPUE³¹</td>
<td>WPUE²¹</td>
<td>WPUE¹¹</td>
<td>WPUE⁰¹</td>
</tr>
<tr>
<td>WPUEF³</td>
<td>WPUEF7</td>
<td>WPUEF6</td>
<td>WPUEF5</td>
<td>WPUEF4</td>
<td>WPUEF³</td>
<td>WPUEF²</td>
<td>WPUEF¹</td>
<td>WPUEF⁰</td>
</tr>
</tbody>
</table>

**Note 1:** If MCLRE = 1, the weak pull-up in RE3 is always enabled; bit WPUE3 is not affected.
**Note 2:** Unimplemented in PIC18(L)F26/27K42.
**Note 3:** Unimplemented in PIC18(L)F26/27/45/46/47K42.
REGISTER 16-6: ODCONx: OPEN-DRAIN CONTROL REGISTER

<table>
<thead>
<tr>
<th>ODCx7</th>
<th>ODCx6</th>
<th>ODCx5</th>
<th>ODCx4</th>
<th>ODCx3</th>
<th>ODCx2</th>
<th>ODCx1</th>
<th>ODCx0</th>
</tr>
</thead>
<tbody>
<tr>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODCONA</td>
<td>ODCA7</td>
<td>ODCA6</td>
<td>ODCA5</td>
<td>ODCA4</td>
<td>ODCA3</td>
<td>ODCA2</td>
<td>ODCA1</td>
<td>ODCA0</td>
</tr>
<tr>
<td>ODCONB</td>
<td>ODCB7</td>
<td>ODCB6</td>
<td>ODCB5</td>
<td>ODCB4</td>
<td>ODCB3</td>
<td>ODCB2</td>
<td>ODCB1</td>
<td>ODCB0</td>
</tr>
<tr>
<td>ODCONC</td>
<td>ODCC7</td>
<td>ODCC6</td>
<td>ODCC5</td>
<td>ODCC4</td>
<td>ODCC3</td>
<td>ODCC2</td>
<td>ODCC1</td>
<td>ODCC0</td>
</tr>
<tr>
<td>ODCOND&lt;1&gt;</td>
<td>ODCD7</td>
<td>ODCD6</td>
<td>ODCD5</td>
<td>ODCD4</td>
<td>ODCD3</td>
<td>ODCD2</td>
<td>ODCD1</td>
<td>ODCD0</td>
</tr>
<tr>
<td>ODCONE&lt;1&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ODCE2</td>
<td>ODCE1</td>
</tr>
<tr>
<td>ODCONF&lt;2&gt;</td>
<td>ODCF7</td>
<td>ODCF6</td>
<td>ODCF5</td>
<td>ODCF4</td>
<td>ODCF3</td>
<td>ODCF2</td>
<td>ODCF1</td>
<td>ODCF0</td>
</tr>
</tbody>
</table>

Note 1: Unimplemented in PIC18(L)F26/27K42.
Note 2: Unimplemented in PIC18(L)F26/27/45/46/47K42.
### REGISTER 16-7: SLRCONx: SLEW RATE CONTROL REGISTER

<table>
<thead>
<tr>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLR7x</td>
<td>SLR6x</td>
<td>SLR5x</td>
<td>SLR4x</td>
<td>SLR3x</td>
<td>SLR2x</td>
<td>SLR1x</td>
<td>SLR0x</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 0</th>
</tr>
</thead>
</table>

**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** \( \text{SLR}_{<7:0>} \): Slew Rate Control on Pins \( \text{Rx}_{<7:0>} \), respectively

- 1 = Port pin slew rate is limited
- 0 = Port pin slews at maximum rate

### TABLE 16-8: SLEW RATE CONTROL REGISTERS

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLRCONA</td>
<td>SLRA7</td>
<td>SLRA6</td>
<td>SLRA5</td>
<td>SLRA4</td>
<td>SLRA3</td>
<td>SLRA2</td>
<td>SLRA1</td>
<td>SLRA0</td>
</tr>
<tr>
<td>SLRCONB</td>
<td>SLRB7</td>
<td>SLRB6</td>
<td>SLRB5</td>
<td>SLRB4</td>
<td>SLRB3</td>
<td>SLRB2</td>
<td>SLRB1</td>
<td>SLRB0</td>
</tr>
<tr>
<td>SLRCONC</td>
<td>SLRC7</td>
<td>SLRC6</td>
<td>SLRC5</td>
<td>SLRC4</td>
<td>SLRC3</td>
<td>SLRC2</td>
<td>SLRC1</td>
<td>SLRC0</td>
</tr>
<tr>
<td>SLRCOND(1)</td>
<td>SLRD7</td>
<td>SLRD6</td>
<td>SLRD5</td>
<td>SLRD4</td>
<td>SLRD3</td>
<td>SLRD2</td>
<td>SLRD1</td>
<td>SLRD0</td>
</tr>
<tr>
<td>SLRCONE(1)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>SLRE2</td>
<td>SLRE1</td>
</tr>
<tr>
<td>SLRCONF(2)</td>
<td>SLRF7</td>
<td>SLRF6</td>
<td>SLRF5</td>
<td>SLRF4</td>
<td>SLRF3</td>
<td>SLRF2</td>
<td>SLRF1</td>
<td>SLRF0</td>
</tr>
</tbody>
</table>

**Note 1:** Unimplemented in PIC18(L)F26/27K42.

**Note 2:** Unimplemented in PIC18(L)F26/27/45/46/47K42.
REGISTER 16-8:  INLVLx: INPUT LEVEL CONTROL REGISTER

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>INLVLx7</td>
<td>INLVLx6</td>
<td>INLVLx5</td>
<td>INLVLx4</td>
<td>INLVLx3</td>
<td>INLVLx2</td>
<td>INLVLx1</td>
<td>INLVLx0</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>INLVL A</td>
<td>INLVL A7</td>
<td>INLVL A6</td>
<td>INLVL A5</td>
<td>INLVL A4</td>
<td>INLVL A3</td>
<td>INLVL A2</td>
<td>INLVL A1</td>
<td>INLVL A0</td>
</tr>
<tr>
<td>INLVL B</td>
<td>INLVL B7</td>
<td>INLVL B6</td>
<td>INLVL B5</td>
<td>INLVL B4</td>
<td>INLVL B3</td>
<td>INLVL B2</td>
<td>INLVL B1</td>
<td>INLVL B0</td>
</tr>
<tr>
<td>INLVL C</td>
<td>INLVL C7</td>
<td>INLVL C6</td>
<td>INLVL C5</td>
<td>INLVL C4</td>
<td>INLVL C3</td>
<td>INLVL C2</td>
<td>INLVL C1</td>
<td>INLVL C0</td>
</tr>
<tr>
<td>INLVL D(2)</td>
<td>INLVL D7</td>
<td>INLVL D6</td>
<td>INLVL D5</td>
<td>INLVL D4</td>
<td>INLVL D3</td>
<td>INLVL D2</td>
<td>INLVL D1</td>
<td>INLVL D0</td>
</tr>
<tr>
<td>INLVL E</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>INLVL E3</td>
</tr>
<tr>
<td>INLVL F(3)</td>
<td>INLVL F7</td>
<td>INLVL F6</td>
<td>INLVL F5</td>
<td>INLVL F4</td>
<td>INLVL F3</td>
<td>INLVL F2</td>
<td>INLVL F1</td>
<td>INLVL F0</td>
</tr>
</tbody>
</table>

Note 1:  Any peripheral using the I²C pins read the I²C ST inputs when enabled via RxyI2C.
2:  Unimplemented in PIC18(L)F26/27K42.
REGISTER 16-9: RxyI2C: I2C PAD Rxy CONTROL REGISTER

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5-4</th>
<th>Bit 3-2</th>
<th>Bit 1-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>U-0</td>
<td>R/W-0/0</td>
<td>SLEW</td>
<td>PU&lt;1:0&gt;</td>
<td>TH&lt;1:0&gt;</td>
</tr>
</tbody>
</table>

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: **Unimplemented**: Read as ‘0’

bit 6: **SLEW**: I2C specific slew rate limiting is enabled
- 1 = I2C specific slew rate limiting is enabled. Standard pad slew limiting is disabled. The SLRxy bit is ignored.
- 0 = Standard GPIO Slew Rate; enabled/disabled via SLRxy bit.

bit 5-4: **PU<1:0>**: I2C Pull-up Selection bits
- 11 = Reserved
- 10 = 10x current of standard weak pull-up
- 01 = 2x current of standard weak pull-up
- 00 = Standard GPIO weak pull-up, enabled via WPUxy bit

bit 3-2: **Unimplemented**: Read as ‘0’

bit 1-0: **TH<1:0>**: I2C Input Threshold Selection bits
- 11 = SMBus 3.0 (1.35 V) input threshold
- 10 = SMBus 2.0 (2.1 V) input threshold
- 01 = I2C specific input thresholds
- 00 = Standard GPIO Input pull-up, enabled via INLVLxy registers

---

TABLE 16-10: I2C PAD CONTROL REGISTERS

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>RB1I2C</td>
<td></td>
<td>SLEW</td>
<td>PU&lt;1:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TH&lt;1:0&gt;</td>
</tr>
<tr>
<td>RB2I2C</td>
<td></td>
<td>SLEW</td>
<td>PU&lt;1:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TH&lt;1:0&gt;</td>
</tr>
<tr>
<td>RC3I2C</td>
<td></td>
<td>SLEW</td>
<td>PU&lt;1:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TH&lt;1:0&gt;</td>
</tr>
<tr>
<td>RC4I2C</td>
<td></td>
<td>SLEW</td>
<td>PU&lt;1:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TH&lt;1:0&gt;</td>
</tr>
<tr>
<td>RD0I2C(f)</td>
<td></td>
<td>SLEW</td>
<td>PU&lt;1:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TH&lt;1:0&gt;</td>
</tr>
<tr>
<td>RD1I2C(f)</td>
<td></td>
<td>SLEW</td>
<td>PU&lt;1:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TH&lt;1:0&gt;</td>
</tr>
</tbody>
</table>

Note 1: Unimplemented in PIC18(L)F26/27K42.
<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>PORTA</td>
<td>RA7</td>
<td>RA6</td>
<td>RA5</td>
<td>RA4</td>
<td>RA3</td>
<td>RA2</td>
<td>RA1</td>
<td>RA0</td>
<td>263</td>
</tr>
<tr>
<td>PORTB</td>
<td>RB7(1)</td>
<td>RB6(1)</td>
<td>RB5</td>
<td>RB4</td>
<td>RB3</td>
<td>RB2</td>
<td>RB1</td>
<td>RB0</td>
<td>263</td>
</tr>
<tr>
<td>PORTC</td>
<td>RC7</td>
<td>RC6</td>
<td>RC5</td>
<td>RC4</td>
<td>RC3</td>
<td>RC2</td>
<td>RC1</td>
<td>RC0</td>
<td>263</td>
</tr>
<tr>
<td>PORTD(6)</td>
<td>RD7</td>
<td>RD6</td>
<td>RD5</td>
<td>RD4</td>
<td>RD3</td>
<td>RD2</td>
<td>RD1</td>
<td>RD0</td>
<td>263</td>
</tr>
<tr>
<td>PORTE</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>RE3</td>
<td>—</td>
<td>—</td>
<td>263</td>
</tr>
<tr>
<td>PORTF(7)</td>
<td>RF7</td>
<td>RF6</td>
<td>RF5</td>
<td>RF4</td>
<td>RF3</td>
<td>RF2</td>
<td>RF1</td>
<td>RF0</td>
<td>263</td>
</tr>
<tr>
<td>TRISA</td>
<td>TRISA7</td>
<td>TRISA6</td>
<td>TRISA5</td>
<td>TRISA4</td>
<td>TRISA3</td>
<td>TRISA2</td>
<td>TRISA1</td>
<td>TRISA0</td>
<td>264</td>
</tr>
<tr>
<td>TRISB</td>
<td>TRISB7(1)</td>
<td>TRISB6(1)</td>
<td>TRISB5</td>
<td>TRISB4</td>
<td>TRISB3</td>
<td>TRISB2</td>
<td>TRISB1</td>
<td>TRISB0</td>
<td>264</td>
</tr>
<tr>
<td>TRISC</td>
<td>TRISC7</td>
<td>TRISC6</td>
<td>TRISC5</td>
<td>TRISC4</td>
<td>TRISC3</td>
<td>TRISC2</td>
<td>TRISC1</td>
<td>TRISC0</td>
<td>264</td>
</tr>
<tr>
<td>TRISD(6)</td>
<td>TRISD7</td>
<td>TRISD6</td>
<td>TRISD5</td>
<td>TRISD4</td>
<td>TRISD3</td>
<td>TRISD2</td>
<td>TRISD1</td>
<td>TRISD0</td>
<td>264</td>
</tr>
<tr>
<td>TRISE(6)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>264</td>
</tr>
<tr>
<td>TRISF(7)</td>
<td>TRISF7</td>
<td>TRISF6</td>
<td>TRISF5</td>
<td>TRISF4</td>
<td>TRISF3</td>
<td>TRISF2</td>
<td>TRISF1</td>
<td>TRISF0</td>
<td>264</td>
</tr>
<tr>
<td>LATA</td>
<td>LATA7</td>
<td>LATA6</td>
<td>LATA5</td>
<td>LATA4</td>
<td>LATA3</td>
<td>LATA2</td>
<td>LATA1</td>
<td>LATA0</td>
<td>264</td>
</tr>
<tr>
<td>LATB</td>
<td>LATB7</td>
<td>LATB6</td>
<td>LATB5</td>
<td>LATB4</td>
<td>LATB3</td>
<td>LATB2</td>
<td>LATB1</td>
<td>LATB0</td>
<td>265</td>
</tr>
<tr>
<td>LACT</td>
<td>LACT7</td>
<td>LACT6</td>
<td>LACT5</td>
<td>LACT4</td>
<td>LACT3</td>
<td>LACT2</td>
<td>LACT1</td>
<td>LACT0</td>
<td>265</td>
</tr>
<tr>
<td>LATD(6)</td>
<td>LATD7</td>
<td>LATD6</td>
<td>LATD5</td>
<td>LATD4</td>
<td>LATD3</td>
<td>LATD2</td>
<td>LATD1</td>
<td>LATD0</td>
<td>265</td>
</tr>
<tr>
<td>LATE(6)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>265</td>
</tr>
<tr>
<td>LATF(7)</td>
<td>LATF7</td>
<td>LATF6</td>
<td>LATF5</td>
<td>LATF4</td>
<td>LATF3</td>
<td>LATF2</td>
<td>LATF1</td>
<td>LATF0</td>
<td>265</td>
</tr>
<tr>
<td>ANSELA</td>
<td>ANSELA7</td>
<td>ANSELA6</td>
<td>ANSELA5</td>
<td>ANSELA4</td>
<td>ANSELA3</td>
<td>ANSELA2</td>
<td>ANSELA1</td>
<td>ANSELA0</td>
<td>266</td>
</tr>
<tr>
<td>ANSELB</td>
<td>ANSELB7</td>
<td>ANSELB6</td>
<td>ANSELB5</td>
<td>ANSELB4</td>
<td>ANSELB3</td>
<td>ANSELB2</td>
<td>ANSELB1</td>
<td>ANSELB0</td>
<td>266</td>
</tr>
<tr>
<td>ANSELC</td>
<td>ANSELC7</td>
<td>ANSELC6</td>
<td>ANSELC5</td>
<td>ANSELC4</td>
<td>ANSELC3</td>
<td>ANSELC2</td>
<td>ANSELC1</td>
<td>ANSELC0</td>
<td>266</td>
</tr>
<tr>
<td>ANSELD(6)</td>
<td>ANSELD7</td>
<td>ANSELD6</td>
<td>ANSELD5</td>
<td>ANSELD4</td>
<td>ANSELD3</td>
<td>ANSELD2</td>
<td>ANSELD1</td>
<td>ANSELD0</td>
<td>266</td>
</tr>
<tr>
<td>ANSELE(6)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>266</td>
</tr>
<tr>
<td>ANSELF(7)</td>
<td>ANSELF7</td>
<td>ANSELF6</td>
<td>ANSELF5</td>
<td>ANSELF4</td>
<td>ANSELF3</td>
<td>ANSELF2</td>
<td>ANSELF1</td>
<td>ANSELF0</td>
<td>266</td>
</tr>
<tr>
<td>WPUA</td>
<td>WPUA7</td>
<td>WPUA6</td>
<td>WPUA5</td>
<td>WPUA4</td>
<td>WPUA3</td>
<td>WPUA2</td>
<td>WPUA1</td>
<td>WPUA0</td>
<td>267</td>
</tr>
<tr>
<td>WPUB</td>
<td>WPUB7</td>
<td>WPUB6</td>
<td>WPUB5</td>
<td>WPUB4</td>
<td>WPUB3</td>
<td>WPUB2</td>
<td>WPUB1</td>
<td>WPUB0</td>
<td>267</td>
</tr>
<tr>
<td>WPUC</td>
<td>WPUC7</td>
<td>WPUC6</td>
<td>WPUC5</td>
<td>WPUC4</td>
<td>WPUC3</td>
<td>WPUC2</td>
<td>WPUC1</td>
<td>WPUC0</td>
<td>267</td>
</tr>
<tr>
<td>WPUD(6)</td>
<td>WPUD7</td>
<td>WPUD6</td>
<td>WPUD5</td>
<td>WPUD4</td>
<td>WPUD3</td>
<td>WPUD2</td>
<td>WPUD1</td>
<td>WPUD0</td>
<td>267</td>
</tr>
<tr>
<td>WPUJE</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>267</td>
</tr>
<tr>
<td>WPUE(6)</td>
<td>WPUE7</td>
<td>WPUE6</td>
<td>WPUE5</td>
<td>WPUE4</td>
<td>WPUE3</td>
<td>WPUE2</td>
<td>WPUE1</td>
<td>WPUE0</td>
<td>267</td>
</tr>
<tr>
<td>ODCONA</td>
<td>ODCNA7</td>
<td>ODCNA6</td>
<td>ODCNA5</td>
<td>ODCNA4</td>
<td>ODCNA3</td>
<td>ODCNA2</td>
<td>ODCNA1</td>
<td>ODCNA0</td>
<td>268</td>
</tr>
<tr>
<td>ODCONB</td>
<td>ODCNB7</td>
<td>ODCNB6</td>
<td>ODCNB5</td>
<td>ODCNB4</td>
<td>ODCNB3</td>
<td>ODCNB2</td>
<td>ODCNB1</td>
<td>ODCNB0</td>
<td>268</td>
</tr>
<tr>
<td>ODCONC</td>
<td>ODCNC7</td>
<td>ODCNC6</td>
<td>ODCNC5</td>
<td>ODCNC4</td>
<td>ODCNC3</td>
<td>ODCNC2</td>
<td>ODCNC1</td>
<td>ODCNC0</td>
<td>268</td>
</tr>
<tr>
<td>ODCOND(6)</td>
<td>ODCOND7</td>
<td>ODCOND6</td>
<td>ODCOND5</td>
<td>ODCOND4</td>
<td>ODCOND3</td>
<td>ODCOND2</td>
<td>ODCOND1</td>
<td>ODCOND0</td>
<td>268</td>
</tr>
<tr>
<td>ODCONE(6)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>268</td>
</tr>
<tr>
<td>ODCONF(7)</td>
<td>ODCF7</td>
<td>ODCF6</td>
<td>ODCF5</td>
<td>ODCF4</td>
<td>ODCF3</td>
<td>ODCF2</td>
<td>ODCF1</td>
<td>ODCF0</td>
<td>268</td>
</tr>
<tr>
<td>SLRCONA</td>
<td>SLRCA7</td>
<td>SLRCA6</td>
<td>SLRCA5</td>
<td>SLRCA4</td>
<td>SLRCA3</td>
<td>SLRCA2</td>
<td>SLRCA1</td>
<td>SLRCA0</td>
<td>269</td>
</tr>
<tr>
<td>SLRCONB</td>
<td>SLRB7</td>
<td>SLRB6</td>
<td>SLRB5</td>
<td>SLRB4</td>
<td>SLRB3</td>
<td>SLRB2</td>
<td>SLRB1</td>
<td>SLRB0</td>
<td>269</td>
</tr>
<tr>
<td>SLRCONC</td>
<td>SLRC7</td>
<td>SLRC6</td>
<td>SLRC5</td>
<td>SLRC4</td>
<td>SLRC3</td>
<td>SLRC2</td>
<td>SLRC1</td>
<td>SLRC0</td>
<td>269</td>
</tr>
<tr>
<td>SLRCOND(6)</td>
<td>SLRCOND7</td>
<td>SLRCOND6</td>
<td>SLRCOND5</td>
<td>SLRCOND4</td>
<td>SLRCOND3</td>
<td>SLRCOND2</td>
<td>SLRCOND1</td>
<td>SLRCOND0</td>
<td>269</td>
</tr>
<tr>
<td>SLRCONE(6)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>269</td>
</tr>
<tr>
<td>SLRCONF(7)</td>
<td>SLRF7</td>
<td>SLRF6</td>
<td>SLRF5</td>
<td>SLRF4</td>
<td>SLRF3</td>
<td>SLRF2</td>
<td>SLRF1</td>
<td>SLRF0</td>
<td>269</td>
</tr>
<tr>
<td>INLVLA</td>
<td>INLVLA7</td>
<td>INLVLA6</td>
<td>INLVLA5</td>
<td>INLVLA4</td>
<td>INLVLA3</td>
<td>INLVLA2</td>
<td>INLVLA1</td>
<td>INLVLA0</td>
<td>270</td>
</tr>
<tr>
<td>INLVB</td>
<td>INLVB7</td>
<td>INLVB6</td>
<td>INLVB5</td>
<td>INLVB4</td>
<td>INLVB3</td>
<td>INLVB2</td>
<td>INLVB1</td>
<td>INLVB0</td>
<td>270</td>
</tr>
</tbody>
</table>

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 I2C pins read the I2C ST inputs when enabled via RxyI2C.
6: Unimplemented in PIC18(L)F26/27K42.
<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>INLVLC</td>
<td>INLVLC7</td>
<td>INLVLC6</td>
<td>INLVLC5</td>
<td>INLVLC4</td>
<td>INLVLC3</td>
<td>INLVLC2</td>
<td>INLVLC1</td>
<td>INLVLC0</td>
<td>270</td>
</tr>
<tr>
<td>INLVLD&lt;sup&gt;(6)&lt;/sup&gt;</td>
<td>INLVLD7</td>
<td>INLVLD6</td>
<td>INLVLD5</td>
<td>INLVLD4</td>
<td>INLVLD3</td>
<td>INLVLD2</td>
<td>INLVLD1&lt;sup&gt;(5)&lt;/sup&gt;</td>
<td>INLVLD0&lt;sup&gt;(5)&lt;/sup&gt;</td>
<td>270</td>
</tr>
<tr>
<td>INLVLF&lt;sup&gt;(7)&lt;/sup&gt;</td>
<td>INLVLF7</td>
<td>INLVLF6</td>
<td>INLVLF5</td>
<td>INLVLF4</td>
<td>INLVLF3</td>
<td>INLVLF2</td>
<td>INLVLF1</td>
<td>INLVLF0</td>
<td>270</td>
</tr>
<tr>
<td>INLVLE</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RB1I2C</td>
<td>—</td>
<td>SLEW</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>TH&lt;1:0&gt;</td>
</tr>
<tr>
<td>RB2I2C</td>
<td>—</td>
<td>SLEW</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>271</td>
</tr>
<tr>
<td>RC3I2C</td>
<td>—</td>
<td>SLEW</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>271</td>
</tr>
<tr>
<td>RC4I2C</td>
<td>—</td>
<td>SLEW</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>271</td>
</tr>
<tr>
<td>RD0I2C&lt;sup&gt;(8)&lt;/sup&gt;</td>
<td>—</td>
<td>SLEW</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>271</td>
</tr>
<tr>
<td>RD1I2C&lt;sup&gt;(8)&lt;/sup&gt;</td>
<td>—</td>
<td>SLEW</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>271</td>
</tr>
</tbody>
</table>

**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 I2C pins read the I2C ST inputs when enabled via RxyI2C.
6. Unimplemented in PIC18(L)F26/27K42.
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:

- I²C

Note: Refer to Table 17-1 for pins that are I²C compatible. Clock and data signals can be routed to any pin, however pins without I²C compatibility will operate at standard TTL/ST logic levels as selected by the INLV 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
MOVF PPSLOCK
MOVLW AAh
MOVF 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
MOVFB PPSLOCK
MOVLB 55h

; Required sequence, next 4 instructions
MOVFB PPSLOCK
MOVFB AAh
MOVFB 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 1. The PPS one-way lock is also removed.
17.8 Register Definitions: PPS Input Selection

REGISTER 17-1: xxxPPS: PERIPHERAL xxx INPUT SELECTION

| 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 = PORTF(2) |
| 100 = PORTE(3) |
| 011 = PORTD(3) |
| 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.
2: Reserved on PIC18LF26/27/45/46/57K42 parts.
3: Reserved on PIC18LF26/27K42 parts.
<table>
<thead>
<tr>
<th>Peripheral</th>
<th>PPS Input Register</th>
<th>Default Pin Selection at POR</th>
<th>Register Reset Value at POR</th>
<th>Input Available from Selected PORx</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWG1</td>
<td>CWG1PPS</td>
<td>RB0</td>
<td>0b0 1000</td>
<td>PIC18(L)F26/27K42: — B C — B C — — — —</td>
</tr>
<tr>
<td>CLCx Input 1</td>
<td>CLCIN0PPS</td>
<td>RA0</td>
<td>0b0 0000</td>
<td>PIC18(L)F26/27K42: — B C — B C — — — —</td>
</tr>
</tbody>
</table>
TABLE 17-1: PPS INPUT REGISTER DETAILS

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>PPS Input Register</th>
<th>Default Pin Selection at POR</th>
<th>Register Reset Value at POR</th>
<th>PIC18(L)F26/27K42</th>
<th>PIC18(L)F45/46/47K42</th>
<th>PIC18(L)F55/56/57K42</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC Conversion</td>
<td>ADACTPPS</td>
<td>RB4</td>
<td>0b0 1100</td>
<td>—</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>Trigger</td>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>B</td>
<td>—</td>
</tr>
<tr>
<td>SPI1 Clock</td>
<td>SPI1SCKPPS</td>
<td>RC3</td>
<td>0b1 0011</td>
<td>—</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>SPI1 Data</td>
<td>SPI1SDIPPS</td>
<td>RC4</td>
<td>0b0 0100</td>
<td>—</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>SPI1 Slave Select</td>
<td>SPI1SSPPS</td>
<td>RA5</td>
<td>0b0 0101</td>
<td>A</td>
<td>—</td>
<td>C</td>
</tr>
<tr>
<td>I2C1 Clock</td>
<td>I2C1SCLPPS</td>
<td>RC3</td>
<td>0b1 0011</td>
<td>—</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>I2C1 Data</td>
<td>I2C1SDAPPS</td>
<td>RC4</td>
<td>0b0 1000</td>
<td>—</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>I2C2 Clock</td>
<td>I2C2SCLPPS</td>
<td>RB1</td>
<td>0b0 1001</td>
<td>—</td>
<td>B</td>
<td>—</td>
</tr>
<tr>
<td>I2C2 Data</td>
<td>I2C2SDAPPS</td>
<td>RB2</td>
<td>0b0 1010</td>
<td>—</td>
<td>B</td>
<td>—</td>
</tr>
<tr>
<td>UART1 Receive</td>
<td>U1RXPPS</td>
<td>RC7</td>
<td>0b0 0111</td>
<td>—</td>
<td>B</td>
<td>—</td>
</tr>
<tr>
<td>UART1 Clear To Send</td>
<td>U1CTSPPS</td>
<td>RC6</td>
<td>0b0 0110</td>
<td>—</td>
<td>B</td>
<td>—</td>
</tr>
<tr>
<td>UART2 Receive</td>
<td>U2RXPPS</td>
<td>RB7</td>
<td>0b0 1111</td>
<td>—</td>
<td>B</td>
<td>—</td>
</tr>
<tr>
<td>UART2 Clear To Send</td>
<td>U2CTSPPS</td>
<td>RB6</td>
<td>0b0 1110</td>
<td>—</td>
<td>B</td>
<td>—</td>
</tr>
</tbody>
</table>
## REGISTER 17-2: RxyPPS: PIN Rxy OUTPUT SOURCE SELECTION REGISTER

<table>
<thead>
<tr>
<th></th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/u</th>
<th>R/W-0/u</th>
<th>R/W-0/u</th>
<th>R/W-0/u</th>
<th>R/W-0/u</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7-6</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>RxyPPS&lt;5:0&gt;</td>
</tr>
</tbody>
</table>

### 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

<table>
<thead>
<tr>
<th>RxyPPS&lt;5:0&gt;</th>
<th>Pin Rxy Output Source</th>
<th>Device Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>PIC18(L)F26/27K42</td>
</tr>
<tr>
<td>0b11 1111</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0b11 0010</td>
<td>ADGRDB</td>
<td>A</td>
</tr>
<tr>
<td>0b11 0000</td>
<td>ADGRDA</td>
<td>A</td>
</tr>
<tr>
<td>0b11 0000</td>
<td>CWG3D</td>
<td>A</td>
</tr>
<tr>
<td>0b10 1111</td>
<td>CWG3C</td>
<td>A</td>
</tr>
<tr>
<td>0b10 1110</td>
<td>CWG3B</td>
<td>A</td>
</tr>
<tr>
<td>0b10 1101</td>
<td>CWG3A</td>
<td>—</td>
</tr>
<tr>
<td>0b10 1100</td>
<td>CWG2D</td>
<td>—</td>
</tr>
<tr>
<td>0b10 1100</td>
<td>CWG2C</td>
<td>—</td>
</tr>
<tr>
<td>0b10 1010</td>
<td>CWG2B</td>
<td>—</td>
</tr>
<tr>
<td>0b10 1001</td>
<td>CWG2A</td>
<td>—</td>
</tr>
<tr>
<td>0b10 1000</td>
<td>DSM1</td>
<td>A</td>
</tr>
<tr>
<td>0b10 0100</td>
<td>PIC2 (SDA)</td>
<td>—</td>
</tr>
<tr>
<td>0b10 0111</td>
<td>PIC2 (SCL)</td>
<td>—</td>
</tr>
<tr>
<td>0b10 0100</td>
<td>PIC1 (SDA)</td>
<td>—</td>
</tr>
<tr>
<td>0b10 0010</td>
<td>PIC1 (SCL)</td>
<td>—</td>
</tr>
<tr>
<td>0b10 0000</td>
<td>SP11 (SS)</td>
<td>A</td>
</tr>
<tr>
<td>0b01 1111</td>
<td>SP11 (SDO)</td>
<td>—</td>
</tr>
<tr>
<td>0b01 1111</td>
<td>SP11 (SCK)</td>
<td>—</td>
</tr>
<tr>
<td>0b01 1100</td>
<td>C2OUT</td>
<td>A</td>
</tr>
<tr>
<td>0b01 1100</td>
<td>C1OUT</td>
<td>A</td>
</tr>
<tr>
<td>0b01 1011</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0b01 1000</td>
<td>UART2 (RTS)</td>
<td>—</td>
</tr>
<tr>
<td>0b01 0111</td>
<td>UART2 (TXDE)</td>
<td>—</td>
</tr>
<tr>
<td>0b01 0110</td>
<td>UART2 (TX)</td>
<td>—</td>
</tr>
<tr>
<td>0b01 0101</td>
<td>UART1 (RTS)</td>
<td>—</td>
</tr>
<tr>
<td>0b01 0100</td>
<td>UART1 (TXDE)</td>
<td>—</td>
</tr>
<tr>
<td>0b01 0011</td>
<td>UART1 (TX)</td>
<td>—</td>
</tr>
</tbody>
</table>
### TABLE 17-2: PPS OUTPUT REGISTER DETAILS

<table>
<thead>
<tr>
<th>RxyPPS&lt;5:0&gt;</th>
<th>Pin Rxy Output Source</th>
<th>Device Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>PIC18(L)F26/27K42</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>PIC18(L)F45/46/47K42</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>PIC18(L)F55/56/57K42</td>
</tr>
</tbody>
</table>

- **Reserved** when not used.
- **A** - Active (output is enabled)
- **B** - Blank (output is disabled)
- **C** - Characteristics
- **D** - Data
- **E** - Enable
- **F** - Function

<table>
<thead>
<tr>
<th>Device Configuration</th>
<th>PIC18(L)F26/27K42</th>
<th>PIC18(L)F45/46/47K42</th>
<th>PIC18(L)F55/56/57K42</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PWM8</strong></td>
<td>A</td>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td><strong>PWM7</strong></td>
<td>A</td>
<td>C</td>
<td>A</td>
</tr>
<tr>
<td><strong>PWM6</strong></td>
<td>A</td>
<td>C</td>
<td>B</td>
</tr>
<tr>
<td><strong>PWM5</strong></td>
<td>A</td>
<td>C</td>
<td>B</td>
</tr>
<tr>
<td><strong>CCP4</strong></td>
<td>B</td>
<td>C</td>
<td>B</td>
</tr>
<tr>
<td><strong>CCP3</strong></td>
<td>B</td>
<td>C</td>
<td>B</td>
</tr>
<tr>
<td><strong>CCP2</strong></td>
<td>B</td>
<td>C</td>
<td>B</td>
</tr>
<tr>
<td><strong>CCP1</strong></td>
<td>B</td>
<td>C</td>
<td>B</td>
</tr>
<tr>
<td><strong>CWG1D</strong></td>
<td>B</td>
<td>C</td>
<td>B</td>
</tr>
<tr>
<td><strong>CWG1C</strong></td>
<td>B</td>
<td>C</td>
<td>B</td>
</tr>
<tr>
<td><strong>CWG1B</strong></td>
<td>B</td>
<td>C</td>
<td>B</td>
</tr>
<tr>
<td><strong>CWG1A</strong></td>
<td>B</td>
<td>C</td>
<td>B</td>
</tr>
<tr>
<td><strong>CLC4OUT</strong></td>
<td>B</td>
<td>C</td>
<td>B</td>
</tr>
<tr>
<td><strong>CLC3OUT</strong></td>
<td>B</td>
<td>C</td>
<td>B</td>
</tr>
<tr>
<td><strong>CLC2OUT</strong></td>
<td>A</td>
<td>C</td>
<td>A</td>
</tr>
<tr>
<td><strong>CLC1OUT</strong></td>
<td>A</td>
<td>C</td>
<td>A</td>
</tr>
<tr>
<td><strong>LATxy</strong></td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
</tbody>
</table>
**REGISTER 17-3: PPSLOCK: PPS LOCK REGISTER**

<table>
<thead>
<tr>
<th>bit 7-1</th>
<th>Unimplemented: Read as '0'</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 0</td>
<td><strong>PPSLOCKED</strong>: PPS Locked bit</td>
</tr>
<tr>
<td>1</td>
<td>= PPS is locked.</td>
</tr>
<tr>
<td>0</td>
<td>= PPS is not locked. PPS selections can be changed.</td>
</tr>
</tbody>
</table>

Legend:
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as '0'
- **u** = Bit is unchanged
- **x** = Bit is unknown
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- -n/n = Value at POR and BOR/Value at all other Resets
## TABLE 17-3: SUMMARY OF REGISTERS ASSOCIATED WITH THE PPS MODULE

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on page</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPSLOCK</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>PPSLOCKED</td>
</tr>
<tr>
<td>INT0PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>INT1PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>INT2PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>T0CKIPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>T1CKIPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>T1GPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>T3CKIPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>T3GPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>T5CKIPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>T5GPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>T2INPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>T4INPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>T6INPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>CCP1PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>CCP2PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>CCP3PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>CCP4PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>SMT1WINPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>SMT1SIGPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>CWG1PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>CWG2PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>CWG3PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>MD1CARLPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>MD1CARHPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>MD1SRCPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>CLCIN0PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>CLCIN1PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>CLCIN2PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>CLCIN3PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>ADACTPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>SPI1SCKPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>SPI1SDIPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>SPI1SSPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>I2C1SCLPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>I2C1SDAPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>I2C2SCLPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>I2C2SDAPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>U1RXPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>U1CTSPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>U2RXPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>U2CTSPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
<tr>
<td>RxYPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>283</td>
</tr>
</tbody>
</table>

**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)F26/27/45/46/47/55/56/57K42 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 PIEx 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)

- IOCANx
- IOCAPx
- Q2
- RAx
- edge detect
- data bus = 0 or 1
- write IOCAFx
- from all other IOCNFx individual pin detectors
- to data bus IOCAFx
- IOC interrupt to CPU core
- IOCIE
18.6 Register Definitions: Interrupt-on-Change Control

**REGISTER 18-1: IOCxP: INTERRUPT-ON-CHANGE POSITIVE EDGE REGISTER EXAMPLE**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>IOCxP&lt;7&gt;: Interrupt-on-Change Positive Edge Enable bit</td>
</tr>
<tr>
<td>0</td>
<td>IOCxP&lt;0&gt;: Interrupt-on-Change Positive Edge Enable bit</td>
</tr>
</tbody>
</table>

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

**REGISTER 18-2: IOCxN: INTERRUPT-ON-CHANGE NEGATIVE EDGE REGISTER EXAMPLE**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>IOCxN&lt;7&gt;: Interrupt-on-Change Negative Edge Enable bit</td>
</tr>
<tr>
<td>0</td>
<td>IOCxN&lt;0&gt;: Interrupt-on-Change Negative Edge Enable bit</td>
</tr>
</tbody>
</table>

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

**REGISTER 18-3: IOCxF: INTERRUPT-ON-CHANGE FLAG REGISTER EXAMPLE**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>IOCxF&lt;7&gt;: Interrupt-on-Change Flag bits</td>
</tr>
<tr>
<td>0</td>
<td>IOCxF&lt;0&gt;: Interrupt-on-Change Flag bits</td>
</tr>
</tbody>
</table>

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 IOCxF[n] = 1 and a positive edge was detected on the IOCx pin, or when IOCxF[n] = 1 and a negative edge was detected on the IOCx pin

0 = No change was detected, or the user cleared the detected change
### TABLE 18-1: IOC REGISTERS

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOCAP</td>
<td>IOCAP7</td>
<td>IOCAP6</td>
<td>IOCAP5</td>
<td>IOCAP4</td>
<td>IOCAP3</td>
<td>IOCAP2</td>
<td>IOCAP1</td>
<td>IOCAP0</td>
</tr>
<tr>
<td>IOCAN</td>
<td>IOCAN7</td>
<td>IOCAN6</td>
<td>IOCAN5</td>
<td>IOCAN4</td>
<td>IOCAN3</td>
<td>IOCAN2</td>
<td>IOCAN1</td>
<td>IOCAN0</td>
</tr>
<tr>
<td>IOCAF</td>
<td>IOCAF7</td>
<td>IOCAF6</td>
<td>IOCAF5</td>
<td>IOCAF4</td>
<td>IOCAF3</td>
<td>IOCAF2</td>
<td>IOCAF1</td>
<td>IOCAF0</td>
</tr>
<tr>
<td>IOCBP</td>
<td>IOCBP7</td>
<td>IOCBP6</td>
<td>IOCBP5</td>
<td>IOCBP4</td>
<td>IOCBP3</td>
<td>IOCBP2</td>
<td>IOCBP1</td>
<td>IOCBP0</td>
</tr>
<tr>
<td>IOCBN</td>
<td>IOCBN7</td>
<td>IOCBN6</td>
<td>IOCBN5</td>
<td>IOCBN4</td>
<td>IOCBN3</td>
<td>IOCBN2</td>
<td>IOCBN1</td>
<td>IOCBN0</td>
</tr>
<tr>
<td>IOCBF</td>
<td>IOCBF7</td>
<td>IOCBF6</td>
<td>IOCBF5</td>
<td>IOCBF4</td>
<td>IOCBF3</td>
<td>IOCBF2</td>
<td>IOCBF1</td>
<td>IOCBF0</td>
</tr>
<tr>
<td>IOCCP</td>
<td>IOCCP7</td>
<td>IOCCP6</td>
<td>IOCCP5</td>
<td>IOCCP4</td>
<td>IOCCP3</td>
<td>IOCCP2</td>
<td>IOCCP1</td>
<td>IOCCP0</td>
</tr>
<tr>
<td>IOCCN</td>
<td>IOCCN7</td>
<td>IOCCN6</td>
<td>IOCCN5</td>
<td>IOCCN4</td>
<td>IOCCN3</td>
<td>IOCCN2</td>
<td>IOCCN1</td>
<td>IOCCN0</td>
</tr>
<tr>
<td>IOCCF</td>
<td>IOCCF7</td>
<td>IOCCF6</td>
<td>IOCCF5</td>
<td>IOCCF4</td>
<td>IOCCF3</td>
<td>IOCCF2</td>
<td>IOCCF1</td>
<td>IOCCF0</td>
</tr>
<tr>
<td>IOCEP</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>IOCEN</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>IOCEF</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOCxF</td>
<td>IOCxF7</td>
<td>IOCxF6</td>
<td>IOCxF5</td>
<td>IOCxF4</td>
<td>IOCxF3</td>
<td>IOCxF2</td>
<td>IOCxF1</td>
<td>IOCxF0</td>
<td>287</td>
</tr>
<tr>
<td>IOCxN</td>
<td>IOCxN7</td>
<td>IOCxN6</td>
<td>IOCxN5</td>
<td>IOCxN4</td>
<td>IOCxN3</td>
<td>IOCxN2</td>
<td>IOCxN1</td>
<td>IOCxN0</td>
<td>287</td>
</tr>
<tr>
<td>IOCxP</td>
<td>IOCxP7</td>
<td>IOCxP6</td>
<td>IOCxP5</td>
<td>IOCxP4</td>
<td>IOCxP3</td>
<td>IOCxP2</td>
<td>IOCxP1</td>
<td>IOCxP0</td>
<td>287</td>
</tr>
</tbody>
</table>

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 PIC18F26/27/45/46/47/55/56/57K42 microcontrollers address 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 16.1, 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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>R = Readable bit</td>
<td>W = Writable bit</td>
<td>U = Unimplemented bit, read as ‘0’</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>SYSCMD: Disable Peripheral System Clock Network bit</td>
<td>FVRMD: Disable Fixed Voltage Reference bit</td>
<td>HLVDMD: Disable High/Low-Voltage Detect bit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>FVRMD: Disable Fixed Voltage Reference bit</td>
<td>HLVDMD: Disable High/Low-Voltage Detect bit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>HLVDMD: Disable High/Low-Voltage Detect bit</td>
<td>CRCMD: Disable CRC Engine bit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>CRCMD: Disable CRC Engine bit</td>
<td>SCANMD: Disable NVM Memory Scanner bit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>SCANMD: Disable NVM Memory Scanner bit</td>
<td>NVMMD: NVM Module Disable bit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>NVMMD: NVM Module Disable bit</td>
<td>CLKRMD: Disable Clock Reference bit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>CLKRMD: Disable Clock Reference bit</td>
<td>IOCMD: Disable Interrupt-on-Change bit, All Ports</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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 µs may be required before accessing data.
## REGISTER 19-2: PMD1: PMD CONTROL REGISTER 1

<table>
<thead>
<tr>
<th>NCO1MD</th>
<th>TMR6MD</th>
<th>TMR5MD</th>
<th>TMR4MD</th>
<th>TMR3MD</th>
<th>TMR2MD</th>
<th>TMR1MD</th>
<th>TMR0MD</th>
</tr>
</thead>
<tbody>
<tr>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 7</th>
<th>NCO1MD: Disable NCO1 Module bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NCO1 module disabled</td>
</tr>
<tr>
<td>0</td>
<td>NCO1 module enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 6</th>
<th>TMR6MD: Disable Timer TMR6 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TMR6 module disabled</td>
</tr>
<tr>
<td>0</td>
<td>TMR6 module enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 5</th>
<th>TMR5MD: Disable Timer TMR5 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TMR5 module disabled</td>
</tr>
<tr>
<td>0</td>
<td>TMR5 module enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 4</th>
<th>TMR4MD: Disable Timer TMR4 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TMR4 module disabled</td>
</tr>
<tr>
<td>0</td>
<td>TMR4 module enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 3</th>
<th>TMR3MD: Disable Timer TMR3 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TMR3 module disabled</td>
</tr>
<tr>
<td>0</td>
<td>TMR3 module enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 2</th>
<th>TMR2MD: Disable Timer TMR2 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TMR2 module disabled</td>
</tr>
<tr>
<td>0</td>
<td>TMR2 module enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 1</th>
<th>TMR1MD: Disable Timer TMR1 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TMR1 module disabled</td>
</tr>
<tr>
<td>0</td>
<td>TMR1 module enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 0</th>
<th>TMR0MD: Disable Timer TMR0 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TMR0 module disabled</td>
</tr>
<tr>
<td>0</td>
<td>TMR0 module enabled</td>
</tr>
</tbody>
</table>

---

**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
## REGISTER 19-3: PMD2: PMD CONTROL REGISTER 2

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Unimplemented</td>
<td>Read as '0'</td>
</tr>
<tr>
<td>6</td>
<td>DACMD: Disable DAC bit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = DAC module disabled</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = DAC module enabled</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>ADCMD: Disable ADCC bit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = ADCC module disabled</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = ADCC module enabled</td>
<td></td>
</tr>
<tr>
<td>4-3</td>
<td>Unimplemented</td>
<td>Read as '0'</td>
</tr>
<tr>
<td>2</td>
<td>CMP2MD: Disable Comparator CMP2 bit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = CMP2 module disabled</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = CMP2 module enabled</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>CMP1MD: Disable Comparator CMP1 bit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = CMP1 module disabled</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = CMP1 module enabled</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>ZCDMD: Disable Zero-Cross Detect module bit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = ZCD module disabled</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = ZCD module enabled</td>
<td></td>
</tr>
</tbody>
</table>

### Legend:
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as '0'
- **u** = Bit is unchanged
- **x** = Bit is unknown
- **'1'** = Bit is set
- **'0'** = Bit is cleared
- **q** = Value depends on condition

### Note 1:
Subject to ZCD bit in CONFIG2H.
## REGISTER 19-4: PMD3: PMD CONTROL REGISTER 3

<table>
<thead>
<tr>
<th>bit</th>
<th>Description</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>PWM8MD: Disable Pulse-Width Modulator PWM8 bit</td>
<td>PWM8 module disabled</td>
<td>PWM8 module enabled</td>
</tr>
<tr>
<td>6</td>
<td>PWM7MD: Disable Pulse-Width Modulator PWM7 bit</td>
<td>PWM7 module disabled</td>
<td>PWM7 module enabled</td>
</tr>
<tr>
<td>5</td>
<td>PWM6MD: Disable Pulse-Width Modulator PWM6 bit</td>
<td>PWM6 module disabled</td>
<td>PWM6 module enabled</td>
</tr>
<tr>
<td>4</td>
<td>PWM5MD: Disable Pulse-Width Modulator PWM5 bit</td>
<td>PWM5 module disabled</td>
<td>PWM5 module enabled</td>
</tr>
<tr>
<td>3</td>
<td>CCP4MD: Disable Capture/Compare/PWM CCP4 bit</td>
<td>CCP4 module disabled</td>
<td>CCP4 module enabled</td>
</tr>
<tr>
<td>2</td>
<td>CCP3MD: Disable Capture/Compare/PWM CCP3 bit</td>
<td>CCP3 module disabled</td>
<td>CCP3 module enabled</td>
</tr>
<tr>
<td>1</td>
<td>CCP2MD: Disable Capture/Compare/PWM CCP2 bit</td>
<td>CCP2 module disabled</td>
<td>CCP2 module enabled</td>
</tr>
<tr>
<td>0</td>
<td>CCP1MD: Disable Capture/Compare/PWM CCP1 bit</td>
<td>CCP1 module disabled</td>
<td>CCP1 module enabled</td>
</tr>
</tbody>
</table>

**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
### REGISTER 19-5: PMD4: PMD CONTROL REGISTER 4

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWG3MD</td>
<td>CWG2MD</td>
<td>CWG1MD</td>
<td>Unimplemented</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWG3MD</td>
<td>CWG2MD</td>
<td>CWG1MD</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

#### 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: PMDS: PMD CONTROL REGISTER 5

<table>
<thead>
<tr>
<th>Bit 7-6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>U-0</td>
<td>U-0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>U-0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
</tr>
<tr>
<td>—</td>
<td>—</td>
<td>U2MD</td>
<td>U1MD</td>
<td>—</td>
<td>SPI1MD</td>
<td>I2C2MD</td>
</tr>
<tr>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>I2C1MD</td>
<td>—</td>
</tr>
</tbody>
</table>

**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 U2MD:** Disable UART2 bit
  - 1 = UART2 module disabled
  - 0 = UART2 module enabled
- **bit 4 U1MD:** Disable UART1 bit
  - 1 = UART1 module disabled
  - 0 = UART1 module enabled
- **bit 3 Unimplemented:** Read as ‘0’
- **bit 2 SPI1MD:** Disable SPI1 Module bit
  - 1 = SPI1 module disabled
  - 0 = SPI1 module enabled
- **bit 1 I2C2MD:** Disable I2C2 Module bit
  - 1 = I2C2 module disabled
  - 0 = I2C2 module enabled
- **bit 0 I2C1MD:** Disable I2C1 Module bit
  - 1 = I2C1 module disabled
  - 0 = I2C1 module enabled
<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>—</td>
<td>SMT1MD</td>
<td>CLC4MD</td>
<td>CLC3MD</td>
<td>CLC2MD</td>
<td>CLC1MD</td>
<td>DSMMD</td>
</tr>
</tbody>
</table>

**Legend:**
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as ‘0’
- **u** = Bit is unchanged
- **x** = Bit is unknown
- **‘1’** = Bit is set
- **‘0’** = Bit is cleared
- **q** = Value depends on condition
- **-n/n** = Value at POR and BOR/Value at all other Resets

**bit 7-6** Unimplemented: Read as ‘0’

**bit 5**
- **SMT1MD**: Disable SMT1 Module bit
  - 1 = SMT1 module disabled
  - 0 = SMT1 module enabled

**bit 4**
- **CLC1MD**: Disable CLC4 Module bit
  - 1 = CLC4 module disabled
  - 0 = CLC4 module enabled

**bit 3**
- **CLC3MD**: Disable CLC3 Module bit
  - 1 = CLC3 module disabled
  - 0 = CLC3 module enabled

**bit 2**
- **CLC2MD**: Disable CLC2 Module bit
  - 1 = CLC2 module disabled
  - 0 = CLC2 module enabled

**bit 1**
- **CLC1MD**: Disable CLC1 Module bit
  - 1 = CLC1 module disabled
  - 0 = CLC1 module enabled

**bit 0**
- **DSMMD**: Disable Data Signal Modulator bit
  - 1 = DSM module disabled
  - 0 = DSM module enabled
REGISTER 19-8: PMD7: PMD CONTROL REGISTER 7

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>DMA2MD</td>
</tr>
</tbody>
</table>

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 7-2 **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

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMD0</td>
<td>SYSCMD</td>
<td>FVRMD</td>
<td>HLVDMD</td>
<td>CRCMD</td>
<td>SCANMD</td>
<td>NVMMD</td>
<td>CLKRMD</td>
<td>IOCMD</td>
<td>290</td>
</tr>
<tr>
<td>PMD1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>291</td>
</tr>
<tr>
<td>PMD2</td>
<td>NCO1MD</td>
<td>TMR6MD</td>
<td>TMR5MD</td>
<td>TMR4MD</td>
<td>TMR3MD</td>
<td>TMR2MD</td>
<td>TMR1MD</td>
<td>TMR0MD</td>
<td>292</td>
</tr>
<tr>
<td>PMD3</td>
<td>PWM8MD</td>
<td>PWM7MD</td>
<td>PWM6MD</td>
<td>PWM5MD</td>
<td>CCP4MD</td>
<td>CCP3MD</td>
<td>CCP2MD</td>
<td>CCP1MD</td>
<td>293</td>
</tr>
<tr>
<td>PMD4</td>
<td>CWG3MD</td>
<td>CWG2MD</td>
<td>CWG1MD</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>294</td>
</tr>
<tr>
<td>PMD5</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>SPI1MD</td>
<td>295</td>
</tr>
<tr>
<td>PMD6</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>I2C2MD</td>
<td>295</td>
</tr>
<tr>
<td>PMD7</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>DMA1MD</td>
<td>297</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>EN</td>
<td>TMR0 Enable bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = The module is enabled and operating</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 = The module is disabled and in the lowest power mode</td>
</tr>
<tr>
<td>6</td>
<td>Unimplemented</td>
<td>Read as '0'</td>
</tr>
<tr>
<td>5</td>
<td>OUT</td>
<td>TMR0 Output bit (read-only)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TMR0 output bit</td>
</tr>
<tr>
<td>4</td>
<td>MD16</td>
<td>TMR0 Operating as 16-Bit Timer Select bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = TMR0 is a 16-bit timer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 = TMR0 is an 8-bit timer</td>
</tr>
<tr>
<td>3-0</td>
<td>OUTPS&lt;3:0&gt;</td>
<td>TMR0 Output Postscaler (Divider) Select bits</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1111 = 1:16 Postscaler</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1110 = 1:15 Postscaler</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1101 = 1:14 Postscaler</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1100 = 1:13 Postscaler</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1011 = 1:12 Postscaler</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1010 = 1:11 Postscaler</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1001 = 1:10 Postscaler</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1000 = 1:9 Postscaler</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0111 = 1:8 Postscaler</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0110 = 1:7 Postscaler</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0101 = 1:6 Postscaler</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0100 = 1:5 Postscaler</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0011 = 1:4 Postscaler</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0010 = 1:3 Postscaler</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001 = 1:2 Postscaler</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0000 = 1:1 Postscaler</td>
</tr>
</tbody>
</table>

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
REGISTER 20-2:  T0CON1: TIMER0 CONTROL REGISTER 1

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS&lt;2:0&gt;</td>
<td>ASYNC</td>
<td>CKPS&lt;3:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Bit</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>TMR0L&lt;7:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>bit 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Bit</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>TMR0H&lt;15:8&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>bit 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 TMR0H<15:8>: TMR0 Counter bits <15:8>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>T0CON0</td>
<td>EN</td>
<td>—</td>
<td>OUT</td>
<td>MD16</td>
<td>OUTPS&lt;3:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td>301</td>
</tr>
<tr>
<td>T0CON1</td>
<td>CS&lt;2:0&gt;</td>
<td>ASYNC</td>
<td>CKPS&lt;3:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>302</td>
</tr>
<tr>
<td>TMR0L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TMR0L&lt;7:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td>303</td>
</tr>
<tr>
<td>TMR0H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TMR0H&lt;15:8&gt;</td>
<td></td>
<td></td>
<td></td>
<td>303</td>
</tr>
</tbody>
</table>

Legend: — = unimplemented location, read as ‘0’. Shaded cells are not used by Timer0.
21.0 TIMER1/3/5 MODULE WITH GATE CONTROL

Timer1/3/5 module is a 16-bit timer/counter with the following features:
- 16-bit timer/counter register pair (TMRxH:TMRxL)
- Programmable internal or external clock source
- 2-bit prescaler
- Dedicated Secondary 32 kHz oscillator circuit
- Optionally synchronized comparator out
- Multiple Timer1/3/5 gate (count enable) sources
- Interrupt on overflow
- Wake-up on overflow (external clock, Asynchronous mode only)

- 16-Bit Read/Write Operation
- Time base for the Capture/Compare function with the CCP modules
- Special Event Trigger (with CCP)
- Selectable Gate Source Polarity
- Gate Toggle mode
- Gate Single-pulse mode
- Gate Value Status
- Gate Event Interrupt

Figure 21-1 is a block diagram of the Timer1/3/5 module.
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.

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 TxCLK register allows 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 (postscaled)
- CMP1/2OUT
- SM1_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.

Table 21-1: Timer1/3/5 Enable Selections

<table>
<thead>
<tr>
<th>ON</th>
<th>GE</th>
<th>Timer1/3/5 Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Count Enabled</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Always On</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Off</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>Off</td>
</tr>
</tbody>
</table>

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.
21.3 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 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 Timer1/3/5 Gate

Timer1/3/5 can be configured to count freely or the count can be enabled and disabled using Timer1/3/5 gate circuitry. This is also referred to as Timer1/3/5 gate enable.

Timer1/3/5 gate can also be driven by multiple selectable sources.

21.6.1 TIMER1/3/5 GATE ENABLE

The Timer1/3/5 Gate Enable mode is enabled by setting the TMRxGE bit of the TxGCON register. The polarity of the Timer1/3/5 Gate Enable mode is configured using the TxGPOL bit of the TxGCON register.

When Timer1/3/5 Gate Enable mode is enabled, Timer1/3/5 will increment on the rising edge of the Timer1/3/5 clock source. When Timer1/3/5 Gate signal is inactive, the timer will not increment and hold the current count. See Figure 21-4 for timing details.

### TABLE 21-2: TIMER1/3/5 GATE ENABLE SELECTIONS

<table>
<thead>
<tr>
<th>TMRxCLK</th>
<th>TxGPOL</th>
<th>TxG</th>
<th>Timer1/3/5 Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>↑</td>
<td>1</td>
<td>1</td>
<td>Counts</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
<td>0</td>
<td>Holds Count</td>
</tr>
<tr>
<td>↑</td>
<td>0</td>
<td>1</td>
<td>Holds Count</td>
</tr>
<tr>
<td>↑</td>
<td>0</td>
<td>0</td>
<td>Counts</td>
</tr>
</tbody>
</table>
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 38.3.1 “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”.

---

**Note:** Enabling Toggle mode at the same time as changing the gate polarity may result in indeterminate operation.
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
• SYNC bit of the TxCON register must be set
• Configure the TMRxCLK register for using secondary oscillator as the clock source
• Enable the SOSCEN 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

TxCKI = 1 when TxTMR Enabled

TxCKI = 0 when TxTMR Enabled

Note 1: Arrows indicate counter increments.

2: In Counter mode, a falling edge must be registered by the counter prior to the first incrementing rising edge of the clock.

FIGURE 21-4: TIMER1/3/5 GATE ENABLE MODE

TMRxGE

TxGPOI

TxG_IN

TxCKI

TxGVAL

Timer1/3/5

N

N + 1

N + 2

N + 3

N + 4
FIGURE 21-5: TIMER1/3/5 GATE TOGGLE MODE

<table>
<thead>
<tr>
<th>TMRxGE</th>
<th>TxGPOI</th>
<th>TxGTM</th>
<th>TxxG_IN</th>
<th>TxCKI</th>
<th>TxGVAL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TIMER1/3/5</th>
<th>N</th>
<th>N + 1</th>
<th>N + 2</th>
<th>N + 3</th>
<th>N + 4</th>
<th>N + 5</th>
<th>N + 6</th>
<th>N + 7</th>
<th>N + 8</th>
</tr>
</thead>
</table>

FIGURE 21-6: TIMER1/3/5 GATE SINGLE-PULSE MODE

<table>
<thead>
<tr>
<th>TMRxGE</th>
<th>TxGPOI</th>
<th>TxGSPM</th>
<th>TxxGGO/</th>
<th>Done</th>
<th>TxG_IN</th>
<th>TxCKI</th>
<th>TxGVAL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Set by software</td>
<td>Cleared by hardware on falling edge of TxGVAL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Counting enabled on rising edge of TxG</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TIMER1/3/5</th>
<th>N</th>
<th>N + 1</th>
<th>N + 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>Bit Name Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timer1</td>
<td>T1</td>
</tr>
<tr>
<td>Timer3</td>
<td>T3</td>
</tr>
<tr>
<td>Timer5</td>
<td>T5</td>
</tr>
</tbody>
</table>

REGISTER 21-1: TXCON: TIMERx CONTROL REGISTER

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 6</th>
<th>bit 5-4</th>
<th>bit 3</th>
<th>bit 2</th>
<th>bit 1</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unimplemented: Read as ‘0’</td>
<td>CKPS&lt;1:0&gt;: Timerx Input Clock Prescale Select bits</td>
<td>Unimplemented: Read as ‘0’</td>
<td>SYNC: Timerx External Clock Input Synchronization Control bit</td>
<td>RD16: 16-Bit Read/Write Mode Enable bit</td>
<td>ON: Timerx On bit</td>
<td></td>
</tr>
</tbody>
</table>

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: TxECON: TIMERx GATE CONTROL REGISTER

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 6</th>
<th>bit 5</th>
<th>bit 4</th>
<th>bit 3</th>
<th>bit 2</th>
<th>bit 1-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>GE</td>
<td>GPOL</td>
<td>GTM</td>
<td>GSPM</td>
<td>GGO/DONE</td>
<td>GVAL</td>
<td>—</td>
</tr>
<tr>
<td>R/W-0/u</td>
<td>R/W-0/u</td>
<td>R/W-0/u</td>
<td>R/W-0/u</td>
<td>R-x</td>
<td>U-0</td>
<td>U-0</td>
</tr>
</tbody>
</table>

### 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
  - **=** 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 TxECON 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

<table>
<thead>
<tr>
<th></th>
<th>R/W-0/u</th>
<th>R/W-0/u</th>
<th>R/W-0/u</th>
<th>R/W-0/u</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CS&lt;4:0&gt;</td>
</tr>
</tbody>
</table>

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 |

<table>
<thead>
<tr>
<th>CS</th>
<th>Timer1 Clock Source</th>
<th>Timer3 Clock Source</th>
<th>Timer5 Clock Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>11111-10001</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>10000</td>
<td>CLC4</td>
<td>CLC4</td>
<td>CLC4</td>
</tr>
<tr>
<td>01111</td>
<td>CLC3</td>
<td>CLC3</td>
<td>CLC3</td>
</tr>
<tr>
<td>01110</td>
<td>CLC2</td>
<td>CLC2</td>
<td>CLC2</td>
</tr>
<tr>
<td>01101</td>
<td>CLC1</td>
<td>CLC1</td>
<td>CLC1</td>
</tr>
<tr>
<td>01100</td>
<td>TMR5 overflow</td>
<td>TMR5 overflow</td>
<td>TMR5 overflow</td>
</tr>
<tr>
<td>01101</td>
<td>TMR3 overflow</td>
<td>Reserved</td>
<td>TMR3 overflow</td>
</tr>
<tr>
<td>01100</td>
<td>Reserved</td>
<td>TMR1 overflow</td>
<td>TMR1 overflow</td>
</tr>
<tr>
<td>01001</td>
<td>TMR0 overflow</td>
<td>TMR0 overflow</td>
<td>TMR0 overflow</td>
</tr>
<tr>
<td>01000</td>
<td>CLKREF</td>
<td>CLKREF</td>
<td>CLKREF</td>
</tr>
<tr>
<td>00111</td>
<td>SOSC</td>
<td>SOSC</td>
<td>SOSC</td>
</tr>
<tr>
<td>00110</td>
<td>MFINTOSC (32 kHz)</td>
<td>MFINTOSC (32 kHz)</td>
<td>MFINTOSC (32 kHz)</td>
</tr>
<tr>
<td>00101</td>
<td>MFINTOSC (500 kHz)</td>
<td>MFINTOSC (500 kHz)</td>
<td>MFINTOSC (500 kHz)</td>
</tr>
<tr>
<td>00100</td>
<td>LFINTOSC</td>
<td>LFINTOSC</td>
<td>LFINTOSC</td>
</tr>
<tr>
<td>00011</td>
<td>HFINTOSC</td>
<td>HFINTOSC</td>
<td>HFINTOSC</td>
</tr>
<tr>
<td>00010</td>
<td>Fosc</td>
<td>Fosc</td>
<td>Fosc</td>
</tr>
<tr>
<td>00001</td>
<td>Fosc/4</td>
<td>Fosc/4</td>
<td>Fosc/4</td>
</tr>
<tr>
<td>00000</td>
<td>T1CKIPPS</td>
<td>T3CKIPPS</td>
<td>T5CKIPPS</td>
</tr>
</tbody>
</table>
**REGISTER 21-4: TxGATE: TIMERx GATE ISM REGISTER**

<table>
<thead>
<tr>
<th></th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/u</th>
<th>R/W-0/u</th>
<th>R/W-0/u</th>
<th>R/W-0/u</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>GSS&lt;4:0&gt;</td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as ‘0’
- ‘-n’ = Value at POR
- ‘+’ = 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**

<table>
<thead>
<tr>
<th>GSS</th>
<th>Timer1 Gate Source</th>
<th>Timer2 Gate Source</th>
<th>Timer3 Gate Source</th>
<th>Timer4 Gate Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>11111-10111 Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>11010</td>
<td>CLC4_out</td>
<td>CLC4_out</td>
<td>CLC4_out</td>
<td></td>
</tr>
<tr>
<td>11001</td>
<td>CLC3_out</td>
<td>CLC3_out</td>
<td>CLC3_out</td>
<td></td>
</tr>
<tr>
<td>11000</td>
<td>CLC2_out</td>
<td>CLC2_out</td>
<td>CLC2_out</td>
<td></td>
</tr>
<tr>
<td>10111</td>
<td>CLC1_out</td>
<td>CLC1_out</td>
<td>CLC1_out</td>
<td></td>
</tr>
<tr>
<td>10110</td>
<td>ZCOOUT</td>
<td>ZCOOUT</td>
<td>ZCOOUT</td>
<td></td>
</tr>
<tr>
<td>10101</td>
<td>CMP2OUT</td>
<td>CMP2OUT</td>
<td>CMP2OUT</td>
<td></td>
</tr>
<tr>
<td>10100</td>
<td>CMP1OUT</td>
<td>CMP1OUT</td>
<td>CMP1OUT</td>
<td></td>
</tr>
<tr>
<td>10011</td>
<td>NCO1OUT</td>
<td>NCO1OUT</td>
<td>NCO1OUT</td>
<td></td>
</tr>
<tr>
<td>10010-10001 Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>10000</td>
<td>PWM8OUT</td>
<td>PWM8OUT</td>
<td>PWM8OUT</td>
<td></td>
</tr>
<tr>
<td>01111</td>
<td>PWM7OUT</td>
<td>PWM7OUT</td>
<td>PWM7OUT</td>
<td></td>
</tr>
<tr>
<td>01110</td>
<td>PWM6OUT</td>
<td>PWM6OUT</td>
<td>PWM6OUT</td>
<td></td>
</tr>
<tr>
<td>01101</td>
<td>PWM5OUT</td>
<td>PWM5OUT</td>
<td>PWM5OUT</td>
<td></td>
</tr>
<tr>
<td>01100</td>
<td>CCP4OUT</td>
<td>CCP4OUT</td>
<td>CCP4OUT</td>
<td></td>
</tr>
<tr>
<td>01011</td>
<td>CCP3OUT</td>
<td>CCP3OUT</td>
<td>CCP3OUT</td>
<td></td>
</tr>
<tr>
<td>01010</td>
<td>CCP2OUT</td>
<td>CCP2OUT</td>
<td>CCP2OUT</td>
<td></td>
</tr>
<tr>
<td>01001</td>
<td>CCP1OUT</td>
<td>CCP1OUT</td>
<td>CCP1OUT</td>
<td></td>
</tr>
<tr>
<td>01000</td>
<td>SMT1_match</td>
<td>SMT1_match</td>
<td>SMT1_match</td>
<td></td>
</tr>
<tr>
<td>00111</td>
<td>TMR6OUT (postscaled)</td>
<td>TMR6OUT (postscaled)</td>
<td>TMR6OUT (postscaled)</td>
<td></td>
</tr>
<tr>
<td>00110</td>
<td>TMR5 overflow</td>
<td>TMR5 overflow</td>
<td>TMR5 overflow</td>
<td></td>
</tr>
<tr>
<td>00101</td>
<td>TMR4OUT (postscaled)</td>
<td>TMR4OUT (postscaled)</td>
<td>TMR4OUT (postscaled)</td>
<td></td>
</tr>
<tr>
<td>00100</td>
<td>TMR3 overflow</td>
<td>TMR3 overflow</td>
<td>TMR3 overflow</td>
<td></td>
</tr>
<tr>
<td>00011</td>
<td>TMR2OUT (postscaled)</td>
<td>TMR2OUT (postscaled)</td>
<td>TMR2OUT (postscaled)</td>
<td></td>
</tr>
<tr>
<td>00010</td>
<td>TMR1 overflow</td>
<td>TMR1 overflow</td>
<td>TMR1 overflow</td>
<td></td>
</tr>
<tr>
<td>00001</td>
<td>TMR0 overflow</td>
<td>TMR0 overflow</td>
<td>TMR0 overflow</td>
<td></td>
</tr>
<tr>
<td>00000</td>
<td>Pin selected by T1GPPS</td>
<td>Pin selected by T3GPPS</td>
<td>Pin selected by T5GPPS</td>
<td></td>
</tr>
</tbody>
</table>
### REGISTER 21-5: TMRxL: TIMERx LOW BYTE REGISTER

<table>
<thead>
<tr>
<th>bit 7-0</th>
<th>TMRxL&lt;7:0&gt;</th>
<th>Timerx Low Byte bits</th>
</tr>
</thead>
</table>

### REGISTER 21-6: TMRxH: TIMERx HIGH BYTE REGISTER

<table>
<thead>
<tr>
<th>bit 7-0</th>
<th>TMRxH&lt;7:0&gt;</th>
<th>Timerx High Byte bits</th>
</tr>
</thead>
</table>

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

- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as ‘0’
- **u** = Bit is unchanged
- **x** = Bit is unknown
- **'1'** = Bit is set
- **'0'** = Bit is cleared
- **-n/n** = Value at POR and BOR/Value at all other Resets
### TABLE 21-3: SUMMARY OF REGISTERS ASSOCIATED WITH TIMER1/3/5 AS A TIMER/COUNTER

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Reset Values on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TxCON</td>
<td>—</td>
<td>—</td>
<td>CKPS&lt;1:0&gt;</td>
<td>—</td>
<td>—</td>
<td>SYNC</td>
<td>RD16</td>
<td>ON</td>
<td>313</td>
</tr>
<tr>
<td>TxGCON</td>
<td>GE</td>
<td>GPOL</td>
<td>GTM</td>
<td>GSPM</td>
<td>GO/DONE</td>
<td>GVAL</td>
<td>—</td>
<td>—</td>
<td>314</td>
</tr>
<tr>
<td>TxCLK</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CS&lt;4:0&gt;</td>
<td>—</td>
<td>—</td>
<td>315</td>
</tr>
<tr>
<td>TxGATE</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>GSS&lt;4:0&gt;</td>
<td>—</td>
<td>—</td>
<td>316</td>
</tr>
<tr>
<td>TMRxL</td>
<td>Least Significant Byte of the 16-bit TMR3 Register</td>
<td>317</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TMRxH</td>
<td>Holding Register for the Most Significant Byte of the 16-bit TMR3 Register</td>
<td>317</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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
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.

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

<table>
<thead>
<tr>
<th>Mode</th>
<th>MODE&lt;4:0&gt;</th>
<th>Output Operation</th>
<th>Operation</th>
<th>Timer Control</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&lt;4:3&gt;</td>
<td></td>
<td>Start</td>
<td>Reset</td>
</tr>
<tr>
<td>00</td>
<td>000</td>
<td>Period Pulse</td>
<td>Software gate (Figure 22-6)</td>
<td>ON = 1</td>
</tr>
<tr>
<td></td>
<td>001</td>
<td></td>
<td>Hardware gate, active-high (Figure 22-7)</td>
<td>ON = 1 &amp; TMRx_ers = 1</td>
</tr>
<tr>
<td></td>
<td>010</td>
<td></td>
<td>Hardware gate, active-low</td>
<td>ON = 1 &amp; TMRx_ers = 0</td>
</tr>
<tr>
<td></td>
<td>011</td>
<td>Period Pulse</td>
<td>Rising or Falling Edge Reset</td>
<td>TMRx_ers ↑</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>Period Pulse</td>
<td>Rising Edge Reset (Figure 22-8)</td>
<td>ON = 1</td>
</tr>
<tr>
<td></td>
<td>101</td>
<td></td>
<td>Falling Edge Reset</td>
<td>TMRx_ers ↑</td>
</tr>
<tr>
<td></td>
<td>110</td>
<td></td>
<td>Low Level Reset</td>
<td>ON = 0</td>
</tr>
<tr>
<td></td>
<td>111</td>
<td></td>
<td>High Level Reset (Figure 22-9)</td>
<td>ON = 1</td>
</tr>
<tr>
<td>01</td>
<td>000</td>
<td>One-Shot</td>
<td>Software Start (Figure 22-10)</td>
<td>ON = 1</td>
</tr>
<tr>
<td></td>
<td>001</td>
<td></td>
<td>Rising Edge Start (Figure 22-9)</td>
<td>ON = 1 &amp; TMRx_ers ↑</td>
</tr>
<tr>
<td></td>
<td>010</td>
<td></td>
<td>Falling Edge Start</td>
<td>ON = 1 &amp; TMRx_ers ↓</td>
</tr>
<tr>
<td></td>
<td>011</td>
<td></td>
<td>Any Edge Start</td>
<td>ON = 1 &amp; TMRx_ers ↑</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>Edge Triggered Start (Note 1)</td>
<td>Rising Edge Start &amp; Rising Edge Reset (Figure 22-12)</td>
<td>ON = 1 &amp; TMRx_ers ↑</td>
</tr>
<tr>
<td></td>
<td>101</td>
<td></td>
<td>Falling Edge Start &amp; Falling Edge Reset</td>
<td>ON = 1 &amp; TMRx_ers ↓</td>
</tr>
<tr>
<td></td>
<td>110</td>
<td></td>
<td>Rising Edge Start &amp; Low Level Reset (Figure 22-13)</td>
<td>ON = 1 &amp; TMRx_ers ↑</td>
</tr>
<tr>
<td></td>
<td>111</td>
<td></td>
<td>Falling Edge Start &amp; High Level Reset</td>
<td>ON = 1 &amp; TMRx_ers ↑</td>
</tr>
<tr>
<td>10</td>
<td>000</td>
<td>Reserved</td>
<td>Reserved</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>001</td>
<td>Edge Triggered Start (Note 1)</td>
<td>Rising Edge Start (Figure 22-12)</td>
<td>ON = 1 &amp; TMRx_ers ↑</td>
</tr>
<tr>
<td></td>
<td>010</td>
<td></td>
<td>Falling Edge Start</td>
<td>ON = 1 &amp; TMRx_ers ↓</td>
</tr>
<tr>
<td></td>
<td>011</td>
<td></td>
<td>Any Edge Start</td>
<td>ON = 1 &amp; TMRx_ers ↑</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>Reserved</td>
<td>Reserved</td>
<td>—</td>
</tr>
<tr>
<td>11</td>
<td>110</td>
<td>Level Triggered Start and Hardware Reset</td>
<td>High Level Start &amp; Low Level Reset (Figure 22-13)</td>
<td>ON = 1 &amp; TMRx_ers = 1</td>
</tr>
<tr>
<td></td>
<td>111</td>
<td></td>
<td>Low Level Start &amp; High Level Reset</td>
<td>ON = 1 &amp; TMRx_ers = 0</td>
</tr>
</tbody>
</table>

**Note:**
1. If ON = 0 then an edge is required to restart the timer after ON = 1.
2. When TTxMR = TTxPR then the next clock clears ON and stops TTxMR at 00h.
3. When TTxMR = TTxPR then the next clock stops TTxMR 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

Note 1: Setting the interrupt flag is synchronized with the instruction clock.
Synchronization may take as many as 2 instruction cycles
2: Cleared by software.
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

Note 1: BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.
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)

Note 1: BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.
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 $\text{MODE}<4:0> = \text{00110}$ will cause the timer to reset on a low level external signal. Selecting $\text{MODE}<4:0> = \text{00111}$ 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)

<table>
<thead>
<tr>
<th>MODE</th>
<th>0b00111</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMRx_clk</td>
<td></td>
</tr>
<tr>
<td>TxPR</td>
<td>5</td>
</tr>
<tr>
<td>Instruction(1)</td>
<td>BSF</td>
</tr>
<tr>
<td>ON</td>
<td></td>
</tr>
<tr>
<td>TMRx_ers</td>
<td></td>
</tr>
<tr>
<td>TxTMR</td>
<td>0 1 2 0 1 2 3 4 5 0</td>
</tr>
<tr>
<td>TMRx_postscaled</td>
<td></td>
</tr>
<tr>
<td>PWM Duty Cycle</td>
<td>3</td>
</tr>
<tr>
<td>PWM Output</td>
<td></td>
</tr>
</tbody>
</table>

Note 1: BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.
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)**

![Software Start One-Shot Mode Timing Diagram](image-url)

Note 1: BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.
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)**

![Timing Diagram](image)

**Note 1**: BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.
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 \( (\text{MODE}<4:0> = 01100) \)
- Falling edge Start and Reset \( (\text{MODE}<4:0> = 01101) \)

The timer resets and clears the ON bit when the timer value matches the T2PR period value. External signal edges will have no effect until after software sets the ON bit. Figure 22-10 illustrates the rising edge hardware limit one-shot operation.

When this mode is used in conjunction with the CCP then the first starting edge trigger, and all subsequent Reset edges, will activate the PWM drive. The PWM drive will deactivate when the timer matches the CCPRx pulse width value and stay deactivated until the timer halts at the T2PR period match unless an external signal edge resets the timer before the match occurs.

![Diagram](image)

**Note 1:** BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.
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 timer is reset and the ON bit is cleared. When the ON bit is cleared by either a T2PR match or by software control a new external signal edge is required after the ON bit is set to start the counter.

When Level Triggered Reset One-Shot mode is used in conjunction with the CCP PWM operation, the PWM drive goes active with the external signal edge that starts the timer. The PWM drive goes inactive when the timer count equals the CCPRx pulse-width count. The PWM drive does not go active when the timer count clears at the T2PR period count match.

FIGURE 22-11: LOW LEVEL RESET, EDGE-TRIGGERED HARDWARE LIMIT ONE-SHOT MODE TIMING DIAGRAM (MODE = 01110)

Note 1: BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.
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 (MODE<4:0> = 10001)
- Falling edge (MODE<4:0> = 10010)
- Rising or Falling edge (MODE<4:0> = 10011)

When an Edge-Triggered Monostable mode is used in conjunction with the CCP PWM operation the PWM drive goes active with the external Reset signal edge that starts the timer, but will not go active when the timer matches the T2PR value. While the timer is incrementing, additional edges on the external Reset signal will not affect the CCP PWM.

FIGURE 22-12: RISING EDGE-TRIGGERED MONOSTABLE MODE TIMING DIAGRAM (MODE = 10001)

Note 1: BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.
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 count, the timer is reset and the ON bit is cleared. When the ON bit is cleared by either a T2PR match or by software control, the timer will stay in Reset until both the ON bit is set and the external signal is not at the Reset level.

When Level Triggered Hardware Limit One-Shot modes are used in conjunction with the CCP PWM operation, the PWM drive goes active with either the external signal edge or the setting of the ON bit, whichever of the two starts the timer.

FIGURE 22-13: LEVEL-TRIGGERED HARDWARE LIMIT ONE-SHOT MODE TIMING DIAGRAM (MODE = 10110)

Note 1: BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.
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

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>Bit Name Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timer2</td>
<td>T2</td>
</tr>
<tr>
<td>Timer4</td>
<td>T4</td>
</tr>
<tr>
<td>Timer6</td>
<td>T6</td>
</tr>
</tbody>
</table>

REGISTER 22-1: TxCLK: TIMERx CLOCK SELECTION REGISTER

<table>
<thead>
<tr>
<th>bit 7-4</th>
<th>bit 3-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS&lt;3:0&gt;</td>
<td></td>
</tr>
<tr>
<td>U-0 U-0 U-0 R/W-0/0 R/W-0/0 R/W-0/0 R/W-0/0</td>
<td>CS&lt;3:0&gt;</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>CS&lt;3:0&gt;</th>
<th>T2TMR Clock Source</th>
<th>TMR4 Clock Source</th>
<th>TMR6 Clock Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>1110</td>
<td>CLC4_out</td>
<td>CLC4_out</td>
<td>CLC4_out</td>
</tr>
<tr>
<td>1111</td>
<td>CLC3_out</td>
<td>CLC3_out</td>
<td>CLC3_out</td>
</tr>
<tr>
<td>1100</td>
<td>CLC2_out</td>
<td>CLC2_out</td>
<td>CLC2_out</td>
</tr>
<tr>
<td>1011</td>
<td>CLC1_out</td>
<td>CLC1_out</td>
<td>CLC1_out</td>
</tr>
<tr>
<td>1010</td>
<td>ZCD_OUT</td>
<td>ZCD_OUT</td>
<td>ZCD_OUT</td>
</tr>
<tr>
<td>1001</td>
<td>NCO1OUT</td>
<td>NCO1OUT</td>
<td>NCO1OUT</td>
</tr>
<tr>
<td>1000</td>
<td>CLKREF_OUT</td>
<td>CLKREF_OUT</td>
<td>CLKREF_OUT</td>
</tr>
<tr>
<td>0111</td>
<td>SOSC</td>
<td>SOSC</td>
<td>SOSC</td>
</tr>
<tr>
<td>0110</td>
<td>MFINTOSC (32 kHz)</td>
<td>MFINTOSC (32 kHz)</td>
<td>MFINTOSC (32 kHz)</td>
</tr>
<tr>
<td>0101</td>
<td>MFINTOSC (500 kHz)</td>
<td>MFINTOSC (500 kHz)</td>
<td>MFINTOSC (500 kHz)</td>
</tr>
<tr>
<td>0100</td>
<td>LFINTOSC</td>
<td>LFINTOSC</td>
<td>LFINTOSC</td>
</tr>
<tr>
<td>0011</td>
<td>HFINTOSC</td>
<td>HFINTOSC</td>
<td>HFINTOSC</td>
</tr>
<tr>
<td>0010</td>
<td>Fosc</td>
<td>Fosc</td>
<td>Fosc</td>
</tr>
<tr>
<td>0001</td>
<td>Fosc/4</td>
<td>Fosc/4</td>
<td>Fosc/4</td>
</tr>
<tr>
<td>0000</td>
<td>Pin selected by T2INPPS</td>
<td>Pin selected by T4INPPS</td>
<td>Pin selected by T6INPPS</td>
</tr>
</tbody>
</table>
REGISTER 22-2: TxA0: TIMER2 EXTERNAL RESET SIGNAL SELECTION REGISTER

<table>
<thead>
<tr>
<th></th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>RSEL&lt;4:0&gt;</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>RSEL&lt;4:0&gt;</th>
<th>T2TMR</th>
<th>TMR4</th>
<th>TMR6</th>
</tr>
</thead>
<tbody>
<tr>
<td>11111-11001</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>11000</td>
<td>UART2_tx_edge</td>
<td>UART2_tx_edge</td>
<td>UART2_tx_edge</td>
</tr>
<tr>
<td>10111</td>
<td>UART2_rx_edge</td>
<td>UART2_rx_edge</td>
<td>UART2_rx_edge</td>
</tr>
<tr>
<td>10110</td>
<td>UART1_tx_edge</td>
<td>UART1_tx_edge</td>
<td>UART1_tx_edge</td>
</tr>
<tr>
<td>10101</td>
<td>UART1_rx_edge</td>
<td>UART1_rx_edge</td>
<td>UART1_rx_edge</td>
</tr>
<tr>
<td>10100</td>
<td>CLC4_out</td>
<td>CLC4_out</td>
<td>CLC4_out</td>
</tr>
<tr>
<td>10011</td>
<td>CLC3_out</td>
<td>CLC3_out</td>
<td>CLC3_out</td>
</tr>
<tr>
<td>10010</td>
<td>CLC2_out</td>
<td>CLC2_out</td>
<td>CLC2_out</td>
</tr>
<tr>
<td>10001</td>
<td>CLC1_out</td>
<td>CLC1_out</td>
<td>CLC1_out</td>
</tr>
<tr>
<td>10000</td>
<td>ZCD_OUT</td>
<td>ZCD_OUT</td>
<td>ZCD_OUT</td>
</tr>
<tr>
<td>01111</td>
<td>CMP2OUT</td>
<td>CMP2OUT</td>
<td>CMP2OUT</td>
</tr>
<tr>
<td>01110</td>
<td>CMP1OUT</td>
<td>CMP1OUT</td>
<td>CMP1OUT</td>
</tr>
<tr>
<td>01101-01100</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>01101</td>
<td>PWM8OUT</td>
<td>PWM8OUT</td>
<td>PWM8OUT</td>
</tr>
<tr>
<td>01100</td>
<td>PWM7OUT</td>
<td>PWM7OUT</td>
<td>PWM7OUT</td>
</tr>
<tr>
<td>01001</td>
<td>PWM6OUT</td>
<td>PWM6OUT</td>
<td>PWM6OUT</td>
</tr>
<tr>
<td>01000</td>
<td>PWM5OUT</td>
<td>PWM5OUT</td>
<td>PWM5OUT</td>
</tr>
<tr>
<td>00111</td>
<td>CCP4OUT</td>
<td>CCP4OUT</td>
<td>CCP4OUT</td>
</tr>
<tr>
<td>00110</td>
<td>CCP3OUT</td>
<td>CCP3OUT</td>
<td>CCP3OUT</td>
</tr>
<tr>
<td>00101</td>
<td>CCP2OUT</td>
<td>CCP2OUT</td>
<td>CCP2OUT</td>
</tr>
<tr>
<td>00100</td>
<td>CCP1OUT</td>
<td>CCP1OUT</td>
<td>CCP1OUT</td>
</tr>
<tr>
<td>00011</td>
<td>TMR6 postscaled</td>
<td>TMR6 postscaled</td>
<td>Reserved</td>
</tr>
<tr>
<td>00010</td>
<td>TMR4 postscaled</td>
<td>Reserved</td>
<td>TMR4 postscaled</td>
</tr>
<tr>
<td>00001</td>
<td>Reserved</td>
<td>T2TMR postscaled</td>
<td>T2TMR postscaled</td>
</tr>
<tr>
<td>00000</td>
<td>Pin selected by T2INPSS</td>
<td>Pin selected by T4INPSS</td>
<td>Pin selected by T6INPSS</td>
</tr>
</tbody>
</table>
## REGISTER 22-3: TxTMR: TIMERx COUNTER REGISTER

<table>
<thead>
<tr>
<th>Bit</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMRx&lt;7:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as ‘0’
- u = Bit is unchanged
- x = Bit is unknown
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared

bit 7-0 **TMRx<7:0>:** Timerx Counter bits

## REGISTER 22-4: TxPR: TIMERx PERIOD REGISTER

<table>
<thead>
<tr>
<th>Bit</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRx&lt;7:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as ‘0’
- u = Bit is unchanged
- x = Bit is unknown
- ‘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

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>CKPS&lt;2:0&gt;</th>
<th>OUTPS&lt;3:0&gt;</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ON</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R/W/HC-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
</tr>
</tbody>
</table>

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

#### bit 7  
**ON**: Timerx On bit(1)
- **1** = Timerx is On
- **0** = Timerx is Off: all counters and state machines are reset

#### bit 6-4  
**CKPS<2:0>**: Timerx-type Clock Prescaler 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**

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSYNC: Timerx Prescaler Synchronization Enable bit(1, 2)</td>
<td>CKPOL: Timerx Clock Polarity Selection bit(3)</td>
<td>CKSYNC: Timerx Clock Synchronization Enable bit(4, 5)</td>
<td>MODE&lt;4:0&gt;: Timerx Control Mode Selection bits(6, 7)</td>
</tr>
</tbody>
</table>

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

**Note 1:** Setting this bit ensures that reading TxTMR will return a valid data value.
**Note 2:** When this bit is ‘1’, Timer2 cannot operate in Sleep mode.
**Note 3:** CKPOL should not be changed while ON = 1.
**Note 4:** Setting this bit ensures glitch-free operation when the ON is enabled or disabled.
**Note 5:** When this bit is set then the timer operation will be delayed by two TxTMR input clocks after the ON bit is set.
**Note 6:** Unless otherwise indicated, all modes start upon ON = 1 and stop upon ON = 0 (stops occur without affecting the value of TxTMR).
**Note 7:** When TxTMR = TxPR, the next clock clears TxTMR, regardless of the operating mode.
TABLE 22-3:  SUMMARY OF REGISTERS ASSOCIATED WITH TIMER2

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TxPR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>320*</td>
</tr>
<tr>
<td>TxTMR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>320*</td>
</tr>
<tr>
<td>TxCON</td>
<td>ON</td>
<td>CKPS&lt;2:0&gt;</td>
<td></td>
<td></td>
<td>OUTPS&lt;3:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td>338</td>
</tr>
<tr>
<td>TxCLK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>335</td>
</tr>
<tr>
<td>TxRST</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RSEL&lt;3:0&gt;</td>
<td>336</td>
</tr>
<tr>
<td>TxHLT</td>
<td>PSYNC</td>
<td>CPOL</td>
<td>CSYNC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>MODE&lt;4:0&gt;</td>
<td>339</td>
</tr>
</tbody>
</table>

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 CCPTMRS 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>
<thead>
<tr>
<th>TABLE 23-1: CCP MODE – TIMER RESOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCP Mode</td>
</tr>
<tr>
<td>Capture</td>
</tr>
<tr>
<td>Compare</td>
</tr>
<tr>
<td>PWM</td>
</tr>
</tbody>
</table>

The assignment of a particular timer to a module is determined by the timer to CCP enable bits in the CCPTMRS 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 23-3) 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 (Fosc/4), or by an external clock source. When Timer1 is clocked by Fosc/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**

---

**FIGURE 23-2: COMPARE MODE OPERATION BLOCK DIAGRAM**

```
+------------------+
| CCPRxH | CCPRxL   |
| Comparator        |
| set CCPxIF        |
| Output Logic      |
| TMR1H | TMR1L   |

+------------------+
| Comparator        |
| Output Logic      |
| TMR1H | TMR1L   |

To Peripherals

PPS

CCPx_out

TRIS Control

RxyPPS

MODE<3:0>

4

S
Q
R

CCPx Pin
```
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.

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.

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 36.2.5 “Auto-Conversion Trigger” for more information.

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 23-4: SIMPLIFIED PWM BLOCK DIAGRAM

Notes:
1. 8-bit timer is concatenated with two bits generated by Fosc or two bits of the internal prescaler to create 10-bit time-base.
2. The alignment of the 10 bits from the CCPR register is determined by the CCPxFMT bit.
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

\[
\text{PWM Period} = \left(\frac{(T2PR) + 1}{4}\right) \cdot \frac{Tos}{(TMR2 \text{ Prescale Value})}
\]

Note 1:  \( Tos = \frac{1}{Fosc} \)

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.

**EQUATION 23-2: PULSE WIDTH**

\[ \text{Pulse Width} = \frac{(\text{CCPRxH:CCPRxL register pair}) \cdot \text{Tosc} \cdot (\text{TMR2 Prescale Value})}{4(T2PR + 1)} \]

**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**

\[ \text{Resolution} = \log\left(\frac{4(T2PR + 1)}{\log(2)}\right) \text{ bits} \]

**Note:** If the pulse-width value is greater than the period, the assigned PWM pin(s) will remain unchanged.
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.

---

**TABLE 23-2: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS (Fosc = 20 MHz)**

<table>
<thead>
<tr>
<th>PWM Frequency</th>
<th>1.22 kHz</th>
<th>4.88 kHz</th>
<th>19.53 kHz</th>
<th>78.12 kHz</th>
<th>156.3 kHz</th>
<th>208.3 kHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timer Prescale</td>
<td>16</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>T2PR Value</td>
<td>0xFF</td>
<td>0xFF</td>
<td>0xFF</td>
<td>0x3F</td>
<td>0x1F</td>
<td>0x17</td>
</tr>
<tr>
<td>Maximum Resolution (bits)</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>8</td>
<td>7</td>
<td>6.6</td>
</tr>
</tbody>
</table>

**TABLE 23-3: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS (Fosc = 8 MHz)**

<table>
<thead>
<tr>
<th>PWM Frequency</th>
<th>1.22 kHz</th>
<th>4.90 kHz</th>
<th>19.61 kHz</th>
<th>76.92 kHz</th>
<th>153.85 kHz</th>
<th>200.0 kHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timer Prescale</td>
<td>16</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>T2PR Value</td>
<td>0x65</td>
<td>0x65</td>
<td>0x65</td>
<td>0x19</td>
<td>0x0C</td>
<td>0x09</td>
</tr>
<tr>
<td>Maximum Resolution (bits)</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>6</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>Bit Name Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCP1</td>
<td>CCP1</td>
</tr>
<tr>
<td>CCP2</td>
<td>CCP2</td>
</tr>
<tr>
<td>CCP3</td>
<td>CCP3</td>
</tr>
<tr>
<td>CCP4</td>
<td>CCP4</td>
</tr>
</tbody>
</table>

REGISTER 23-1: CCPxCON: CCPx CONTROL REGISTER

<table>
<thead>
<tr>
<th>EN</th>
<th>—</th>
<th>OUT</th>
<th>FMT</th>
<th>MODE&lt;3:0&gt;</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td>—</td>
<td>OUT</td>
<td>FMT</td>
<td>MODE&lt;3:0&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>MODE</th>
<th>Operating Mode</th>
<th>Operation</th>
<th>Set CCPxIF</th>
</tr>
</thead>
<tbody>
<tr>
<td>11xx</td>
<td>PWM</td>
<td>PWM operation</td>
<td>Yes</td>
</tr>
<tr>
<td>1011</td>
<td>Compare</td>
<td>Pulse output; clear TMR1(^{(2)})</td>
<td>Yes</td>
</tr>
<tr>
<td>1010</td>
<td>Compare</td>
<td>Pulse output</td>
<td>Yes</td>
</tr>
<tr>
<td>1001</td>
<td>Compare</td>
<td>Clear output(^{(1)})</td>
<td>Yes</td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td>Set output(^{(1)})</td>
<td>Yes</td>
</tr>
<tr>
<td>0111</td>
<td>Capture</td>
<td>Every 16th rising edge of CCPx input</td>
<td>Yes</td>
</tr>
<tr>
<td>0110</td>
<td>Capture</td>
<td>Every 4th rising edge of CCPx input</td>
<td>Yes</td>
</tr>
<tr>
<td>0101</td>
<td>Capture</td>
<td>Every rising edge of CCPx input</td>
<td>Yes</td>
</tr>
<tr>
<td>0100</td>
<td>Capture</td>
<td>Every falling edge of CCPx input</td>
<td>Yes</td>
</tr>
<tr>
<td>0011</td>
<td></td>
<td>Every edge of CCPx input</td>
<td>Yes</td>
</tr>
<tr>
<td>0010</td>
<td>Compare</td>
<td>Toggle output</td>
<td>Yes</td>
</tr>
<tr>
<td>0001</td>
<td></td>
<td>Toggle output; clear TMR1(^{(2)})</td>
<td>Yes</td>
</tr>
<tr>
<td>0000</td>
<td>Disabled</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

**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**

<table>
<thead>
<tr>
<th>bit 7-6</th>
<th>bit 5-4</th>
<th>bit 3-2</th>
<th>bit 1-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>C4TSEL&lt;1:0&gt;</td>
<td>C3TSEL&lt;1:0&gt;</td>
<td>C2TSEL&lt;1:0&gt;</td>
<td>C1TSEL&lt;1:0&gt;</td>
</tr>
</tbody>
</table>

**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: CCPxCAP: CAPTURE INPUT SELECTION multiplexer register**

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/x</th>
<th>R/W-0/x</th>
<th>R/W-0/x</th>
<th>CTS&lt;2:0&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>bit 7</td>
</tr>
</tbody>
</table>

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 2-0 **CTS<2:0>: Capture Trigger Input Selection bits**

<table>
<thead>
<tr>
<th>CTS&lt;1:0&gt;</th>
<th>Connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>CLC4_out</td>
</tr>
<tr>
<td>110</td>
<td>CLC3_out</td>
</tr>
<tr>
<td>101</td>
<td>CLC2_out</td>
</tr>
<tr>
<td>100</td>
<td>CLC1_out</td>
</tr>
<tr>
<td>011</td>
<td>IOC_interrupt</td>
</tr>
<tr>
<td>010</td>
<td>CMP2_output</td>
</tr>
<tr>
<td>001</td>
<td>CMP1_output</td>
</tr>
<tr>
<td>000</td>
<td>Pin selected by CCP1PPS</td>
</tr>
</tbody>
</table>

**REGISTER 23-4: CCPRxL: CCPx REGISTER LOW BYTE**

<table>
<thead>
<tr>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
</tr>
</thead>
<tbody>
<tr>
<td>RL&lt;7:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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<5:0>:** CCPW<1:0> – Pulse-Width LS 2 bits
- **RL<5:0>:** Not used
REGISTER 23-5: CCPRxH: CCPx REGISTER HIGH BYTE

<table>
<thead>
<tr>
<th>Bit 7-0</th>
<th>MODE = Capture Mode:</th>
</tr>
</thead>
<tbody>
<tr>
<td>RH&lt;7:0&gt;</td>
<td>MSB of captured TMR1 value</td>
</tr>
<tr>
<td>MODE = Compare Mode:</td>
<td></td>
</tr>
<tr>
<td>RH&lt;7:0&gt;</td>
<td>MSB compared to TMR1 value</td>
</tr>
<tr>
<td>MODE = PWM Mode &amp;&amp; FMT = 0:</td>
<td></td>
</tr>
<tr>
<td>RH&lt;7:2&gt;</td>
<td>Not used</td>
</tr>
<tr>
<td>RH&lt;1:0&gt;</td>
<td>CCPW&lt;9:2&gt; – Pulse-Width MS 2 bits</td>
</tr>
<tr>
<td>MODE = PWM Mode &amp;&amp; FMT = 1:</td>
<td></td>
</tr>
<tr>
<td>RH&lt;7:0&gt;</td>
<td>CCPW&lt;9:2&gt; – Pulse-Width MS 8 bits</td>
</tr>
</tbody>
</table>

TABLE 23-4: SUMMARY OF REGISTERS ASSOCIATED WITH CCPx

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCPxCON</td>
<td>EN</td>
<td>—</td>
<td>OUT</td>
<td>FMT</td>
<td>—</td>
<td>MODE&lt;3:0&gt;</td>
<td></td>
<td></td>
<td>350</td>
</tr>
<tr>
<td>CCPxCAP</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CTS&lt;1:0&gt;</td>
<td></td>
<td>352</td>
</tr>
<tr>
<td>CCPRxL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CCPRx&lt;7:0&gt;</td>
<td></td>
<td>352</td>
</tr>
<tr>
<td>CCPRxH</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CCPRx&lt;15:8&gt;</td>
<td></td>
<td>353</td>
</tr>
<tr>
<td>CCPTMR0</td>
<td>C4TSEL&lt;1:0&gt;</td>
<td>C3TSEL&lt;1:0&gt;</td>
<td>C2TSEL&lt;1:0&gt;</td>
<td>C1TSEL&lt;1:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>351</td>
</tr>
</tbody>
</table>

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
- PWMxCCON

Note: The corresponding TRIS bit must be cleared to enable the PWM output on the PWMx pin.

FIGURE 24-1: SIMPLIFIED PWM BLOCK DIAGRAM

![Simplied PWM Block Diagram](image)

Note 1: 8-bit timer is concatenated with two bits generated by Fosc or two bits of the internal prescaler to create 10-bit time-base.

FIGURE 24-2: PWM OUTPUT

![PWM Output Diagram](image)

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”.

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 CCPTMRSS 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.
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 PWMxDCL (8 MSb) and PWMxDCH<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**

\[
PWM\ Period = \frac{(T2PR + 1) \times \frac{1}{\text{Tosc}}}{(\text{TMR2 Prescale Value})}
\]

\[
\text{Note: } \frac{1}{\text{Tosc}} = \frac{1}{\text{Fosc}}
\]

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**

\[
Pulse\ Width = \frac{(\text{PWMxDCH}:\text{PWMxDCL}<7:6>) \times \text{Tosc}}{(\text{TMR2 Prescale Value})}
\]

\[
\text{Note: } \frac{1}{\text{Tosc}} = \frac{1}{\text{Fosc}}
\]

**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

\[
\text{Resolution} = \frac{\log(4(T2PR + 1))}{\log(2)} \text{ bits}
\]

Note: If the pulse-width value is greater than the period, the assigned PWM pin(s) will remain unchanged.

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 Fosc/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 Fosc/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.

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>Bit Name Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWM3</td>
<td>PWM3</td>
</tr>
<tr>
<td>PWM4</td>
<td>PWM4</td>
</tr>
</tbody>
</table>

**REGISTER 24-1: PWMxCON: PWM CONTROL REGISTER**

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>U-0</th>
<th>R-0/0</th>
<th>R/W-0/0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td>—</td>
<td>bit 0</td>
<td>OUT</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-1/1</th>
<th>R/W-0/0</th>
<th>R/W-1/1</th>
<th>R/W-0/0</th>
<th>R/W-1/1</th>
<th>R/W-0/0</th>
<th>R/W-1/1</th>
</tr>
</thead>
<tbody>
<tr>
<td>P8TSEL&lt;1:0&gt;</td>
<td>P7TSEL&lt;1:0&gt;</td>
<td>P6TSEL&lt;1:0&gt;</td>
<td>P5TSEL&lt;1:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 7</td>
<td>bit 6</td>
<td>bit 5</td>
<td>bit 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 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

<table>
<thead>
<tr>
<th>Bit 7-0</th>
<th>DC&lt;9:2&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
</tr>
<tr>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
</tr>
<tr>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
</tr>
<tr>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
</tr>
</tbody>
</table>

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

**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

<table>
<thead>
<tr>
<th>Bit 7-6</th>
<th>DC&lt;1:0&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>R/W-x/u</td>
<td>U-0</td>
</tr>
<tr>
<td>R/W-x/u</td>
<td>U-0</td>
</tr>
<tr>
<td>U-0</td>
<td>U-0</td>
</tr>
<tr>
<td>U-0</td>
<td>U-0</td>
</tr>
</tbody>
</table>

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

**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

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWMxCON</td>
<td>EN</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>358</td>
</tr>
<tr>
<td>PWMxDCH</td>
<td></td>
<td>DC&lt;9:2&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>360</td>
</tr>
<tr>
<td>PWMxDCL</td>
<td>DC&lt;1:0&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>360</td>
</tr>
<tr>
<td>CCPTMRS1</td>
<td>P8TSEL&lt;1:0&gt;</td>
<td>P7TSEL&lt;1:0&gt;</td>
<td>P6TSEL&lt;1:0&gt;</td>
<td>P5TSEL&lt;1:0&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>359</td>
</tr>
</tbody>
</table>

Legend:  
- **-** = Unimplemented locations, read as ‘0’, **u** = unchanged, **x** = unknown. Shaded cells are not used by the PWM.
25.0 SIGNAL MEASUREMENT TIMER (SMT)

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 (SMT1L/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, SMT1TMR 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:
- \( F_{\text{OSC}} \)
- \( F_{\text{OSC}}/4 \)
- \( F_{\text{INTOSC}} \) 16 MHz
- \( F_{\text{LINTOSC}} \)
- \( F_{\text{MFINTOSC}} \) 31.25 kHz
The SMT clock source is selected by configuring the CSEL<2:0> bits in the SMT1CLK register. The clock source can also be prescaled using the PS<1:0> bits of the SMT1CON0 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 SMT1TMR rolls over to '0'. This happens when SMT1TMR = SMT1PR, regardless of mode. Hence, in any mode that relies on an external signal or a window to reset the timer, proper operation requires that SMT1PR be set to a period larger than that of the expected signal or window.

25.2 Basic Timer Function Registers
The SMT1TMR time base and the SMT1CPW/SMT1PR/SMT1CPR buffer registers serve several functions and can be manually updated using software.

25.2.1 TIME BASE
The SMT1TMR 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 SMT1STAT register. It can be written to and read from software, but it is not guarded for atomic access, therefore reads and writes to the SMT1TMR should only be made when the GO = 0, or the software should have other measures to ensure integrity of SMT1TMR reads/writes.

25.2.2 PULSE-WIDTH LATCH REGISTERS
The SMT1CPW registers are the 24-bit SMT pulse-width latch. They are used to latch in the value of the SMT1TMR when triggered by various signals, which are determined by the mode the SMT is currently in. The SMT1CPW registers can also be updated with the current value of the SMT1TMR value by setting the CPWUP bit of the SMT1STAT register.

25.2.3 PERIOD LATCH REGISTERS
The SMT1CPR registers are the 24-bit SMT period latch. They are used to latch in other values of the SMT1TMR when triggered by various other signals, which are determined by the mode the SMT is currently in.

The SMT1CPR registers can also be updated with the current value of the SMT1TMR value by setting the CPRUP bit in the SMT1STAT register.

25.3 Halt Operation
The counter can be prevented from rolling-over using the STP bit in the SMT1CON0 register. When halting is enabled, the period match interrupt persists until the SMT1TMR is reset (either by a manual Reset, Section 25.2.1 "Time Base") or by clearing the GO bit of the SMT1CON1 register and writing the SMT1TMR 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 SMT1CON0 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 SMT1STAT 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 SMT1STAT 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 SMT1STAT 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 SMT1TMR 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 (SMT1TMR = SMT1PR), SMT1TMR is reset and the period match interrupt trips. See Figure 25-3.

<table>
<thead>
<tr>
<th>MODE</th>
<th>Mode of Operation</th>
<th>Synchronous Operation</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>Timer</td>
<td>Yes</td>
<td>Section 25.6.1 “Timer Mode”</td>
</tr>
<tr>
<td>0001</td>
<td>Gated Timer</td>
<td>Yes</td>
<td>Section 25.6.2 “Gated Timer Mode”</td>
</tr>
<tr>
<td>0010</td>
<td>Period and Duty Cycle Acquisition</td>
<td>Yes</td>
<td>Section 25.6.3 “Period and Duty Cycle Mode”</td>
</tr>
<tr>
<td>0011</td>
<td>High and Low Time Measurement</td>
<td>Yes</td>
<td>Section 25.6.4 “High and Low Measure Mode”</td>
</tr>
<tr>
<td>0100</td>
<td>Windowed Measurement</td>
<td>Yes</td>
<td>Section 25.6.5 “Windowed Measure Mode”</td>
</tr>
<tr>
<td>0101</td>
<td>Gated Windowed Measurement</td>
<td>Yes</td>
<td>Section 25.6.6 “Gated Windowed Measure Mode”</td>
</tr>
<tr>
<td>0110</td>
<td>Time of Flight</td>
<td>Yes</td>
<td>Section 25.6.7 “Time of Flight Measure Mode”</td>
</tr>
<tr>
<td>0111</td>
<td>Capture</td>
<td>Yes</td>
<td>Section 25.6.8 “Capture Mode”</td>
</tr>
<tr>
<td>1000</td>
<td>Counter</td>
<td>No</td>
<td>Section 25.6.9 “Counter Mode”</td>
</tr>
<tr>
<td>1001</td>
<td>Gated Counter</td>
<td>No</td>
<td>Section 25.6.10 “Gated Counter Mode”</td>
</tr>
<tr>
<td>1010</td>
<td>Windowed Counter</td>
<td>No</td>
<td>Section 25.6.11 “Windowed Counter Mode”</td>
</tr>
<tr>
<td>1011-1111 Reserved</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
</tbody>
</table>
FIGURE 25-3: TIMER MODE TIMING DIAGRAM

- SMTx Clock
- SMTxEN
- SMTxGO
- SMTxGO sync
- SMTxPR
- SMTxTMR
- SMTxF
25.6.2 GATED TIMER MODE

Gated Timer mode uses the SMTSIGx input to control whether or not the SMT1TMR will increment. Upon a falling edge of the external signal, the SMT1CPW register will update to the current value of the SMT1TMR. 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

- SMTx_signal
- SMTx_signalsync
- SMTx_Clock
- SMTxEN
- SMTxGO
- SMTxGO_sync
- SMTxPR: 0xFFFFF
- SMTxTMR: 0 1 2 3 4 5 6 7
- SMTxCPW: 5 7
- SMTxPWAI
FIGURE 25-5: GATED TIMER MODE SINGLE ACQUISITION TIMING DIAGRAM

- **SMTx_signal**
- **SMTx_signalsync**
- **SMTx_Clock**
- **SMTxEN**
- **SMTxGO**
- **SMTxGO_sync**
- **SMTxPR**
- **SMTxTMR**
- **SMTxCPW**
- **SMTxPWAIF**

Timing diagram showing the sequence of events for each signal.
25.6.3 PERIOD AND DUTY CYCLE MODE

In Duty Cycle mode, either the duty cycle or period (depending on polarity) of the SMT1_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 SMT1TMR 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

SMTx_signal
SMTx_signalsync
SMTx_Clock
SMTxEN
SMTxGO
SMTxGO_sync
SMTxTMR
SMTxCPW
SMTxCPR
SMTxPWAIF
SMTxPRAIF
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 SMT1TMR on a rising edge on the SMTSIGx input, then updates the SMT1CPW register with the value and resets the SMT1TMR on a falling edge, starting to increment again. Upon observing another rising edge, it updates the SMT1CPR register with its current value and once again resets the SMT1TMR 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

SMTx_signal

SMTx_signalsync

SMTx_Clock

SMTx_EN

SMTx_GO

SMTx_GO_sync

SMTx_TMR

SMTx_CPW

SMTx_CPR

SMTx_PWAIF

SMTx_PRAIF
FIGURE 25-9: HIGH AND LOW MEASURE MODE SINGLE ACQUISITION TIMING DIAGRAM

SMTx_signal
SMTx_signalSync
SMTx_SignalSync
SMTx_CLK
SMTx_EN
SMTx_GO
SMTx_GO_sync
SMTx_TMR

0
1 2 3 4 5 1 2 3 4 5 6
SMTx_CPW
SMTx_CPR
SMTx_PWAIF
SMTx_PRAIF
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 SMT1CPR 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

SMTxWIN
SMTxWIN_sync
SMTxClock
SMTxEN
SMTxGO
SMTxGO_sync
SMTxTMR
0 1 2 3 4 5 6 7 8 9 10 11 12 1 2 3 4 5 6 7 8 1 2 3 4
SMTxCPR
SMTxPRAIF

12 8
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 SMT1_signal input over a known input window. It does so by incrementing the timer on each pulse of the clock signal while the SMT1_signal input is high, updating the SMT1CPR 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

SMTxWIN
SMTxWIN_sync
SMTx_signal
SMTx_signalsync
SMTx Clock
SMTxEN
SMTxGO
SMTxGO_sync
SMTxTMR
SMTxCPR
SMTxPRAIF
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 SMT1_signal input, beginning to increment the timer upon observing a rising edge on the SMTWINx input, while updating the SMT1CPR register and resetting the timer upon observing a rising edge on the SMT1_signal input. In the event of two SMTWINx rising edges without an SMT1_signal rising edge, it will update the SMT1CPW 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

- SMTxWIN
- SMTxWIN_sync
- SMTx_signal
- SMTx_signalsync
- SMTx Clock
- SMTxEN
- SMTxGO
- SMTxGO_sync
- SMTxTMR
- SMTxCPW
- SMTxCPR
- SMTxPWAIF
- SMTxPRAIF
FIGURE 25-15: TIME OF FLIGHT MODE SINGLE ACQUISITION TIMING DIAGRAM

SMTxWIN

SMTxWIN_sync

SMTx_signal

SMTx_signalsync

SMTx Clock

SMTxEN

SMTxGO

SMTxGO_sync

SMTxTMR: 0, 1, 2, 3, 4, 5

SMTxCPW

SMTxCPR

SMTxPWAIF

SMTxPRAIF
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 SMT1CPR 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**

- SMTxWIN
- SMTxWIN_sync
- SMTx Clock
- SMTxEN
- SMTxGO
- SMTxGO_sync
- SMTxTMR: 0 1 2 3
- SMTxCPR
- SMTxPWAIF
- SMTxPRAIF
25.6.9 COUNTER MODE

This mode increments the timer on each pulse of the SMT1_signal input. This mode is asynchronous to the SMT clock and uses the SMT1_signal as a time source. The SMT1CPW register will be updated with the current SMT1TMR value on the rising edge of the SMT1WIN input. See Figure 25-18.

FIGURE 25-18: COUNTER MODE TIMING DIAGRAM
25.6.10 GATED COUNTER MODE

This mode counts pulses on the SMT1_signal input, gated by the SMT1WIN input. It begins incrementing the timer upon seeing a rising edge of the SMT1WIN input and updates the SMT1CPW register upon a falling edge on the SMT1WIN 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 SMT1_signal input, within a window dictated by the SMT1WIN input. It begins counting upon seeing a rising edge of the SMT1WIN input, updates the SMT1CPW register on a falling edge of the SMT1WIN input, and updates the SMT1CPR register on each rising edge of the SMT1WIN 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 SMT1CPW and SMT1CPR registers, the circumstances for which are dependent on the SMT mode, and are discussed in each mode’s specific section. The SMT1CPW interrupt is controlled by SMT1PWAIF and SMT1PWAIE bits in the respective PIR and PIE registers. The SMT1CPR interrupt is controlled by the SMT1PR AIF and SMT1PR AIE bits, also located in the respective PIR and PIE registers.

In synchronous SMT modes, the interrupt trigger is synchronized to the SMT1CLK. 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 SMT1TMR, matching SMT1PR with its period match limit functionality described in Section 25.3 “Halt Operation”. The period match interrupt is controlled by SMT1IF and SMT1IE, 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**

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>Bit Name Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMT1</td>
<td>SMT1</td>
</tr>
</tbody>
</table>

**REGISTER 25-1: SMT1CON0: SMT CONTROL REGISTER 0**

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>PS&lt;1:0&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN&lt;1&gt;</td>
<td>—</td>
<td>STP</td>
<td>WPOL</td>
<td>SPOL</td>
<td>CPOL</td>
<td>PS&lt;1:0&gt;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**

R = Readable bit  
W = Writable bit  
U = Unimplemented bit, read as ‘0’  
u = Bit is unchanged  
x = Bit is unknown  
‘1’ = Bit is set  
‘0’ = Bit is cleared

- **bit 7 EN**: SMT Enable bit(1)
  - 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 SMT1TMR = SMT1PR:
    - 1 = Counter remains SMT1PR; period match interrupt occurs when clocked
    - 0 = Counter resets to 24'h0000000; period match interrupt occurs when clocked

- **bit 4 WPOL**: SMT1WIN Input Polarity Control bit
  - 1 = SMT1WIN signal is active-low/falling edge enabled  
  - 0 = SMT1WIN signal is active-high/rising edge enabled

- **bit 3 SPOL**: SMT1SIG Input Polarity Control bit
  - 1 = SMT1_signal is active-low/falling edge enabled  
  - 0 = SMT1_signal is active-high/rising edge enabled

- **bit 2 CPOL**: SMT Clock Input Polarity Control bit
  - 1 = SMT1TMR increments on the falling edge of the selected clock signal  
  - 0 = SMT1TMR 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: SMT1CON1: SMT CONTROL REGISTER 1

<table>
<thead>
<tr>
<th>R/W/HC&lt;0/0&gt;</th>
<th>R/W&lt;0/0&gt;</th>
<th>U&lt;0&gt;</th>
<th>U&lt;0&gt;</th>
<th>R/W&lt;0/0&gt;</th>
<th>R/W&lt;0/0&gt;</th>
<th>R/W&lt;0/0&gt;</th>
<th>R/W&lt;0/0&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>GO</td>
<td>REPEAT</td>
<td>—</td>
<td>—</td>
<td>MODE&lt;3:0&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 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
- **1110**: 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: SMT1STAT: SMT STATUS REGISTER

<table>
<thead>
<tr>
<th></th>
<th>R/W/HC-0/0</th>
<th>R/W/HC-0/0</th>
<th>R/W/HC-0/0</th>
<th>U-0</th>
<th>U-0</th>
<th>R-0/0</th>
<th>R-0/0</th>
<th>R-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPRUP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPWUP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RST</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
<td></td>
<td>TS</td>
<td>WS</td>
<td>AS</td>
</tr>
</tbody>
</table>

### 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 SMT1PRx registers
  - 0 = SMT1PRx registers update is complete

- **bit 6**: **CPWUP**: SMT Manual Pulse Width Buffer Update bit
  - 1 = Request update to SMT1CPW registers
  - 0 = SMT1CPW registers update is complete

- **bit 5**: **RST**: SMT Manual Timer Reset bit
  - 1 = Request Reset to SMT1TMR registers
  - 0 = SMT1TMR 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**: SMT1WIN 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: SMT1CLK: SMT CLOCK SELECTION REGISTER

<table>
<thead>
<tr>
<th>bits</th>
<th>description</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7-3</td>
<td>Unimplemented</td>
<td>Read as '0'</td>
</tr>
<tr>
<td>bit 2-0</td>
<td>CSEL&lt;2:0&gt;</td>
<td>SMT Clock Selection bits</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>value</th>
<th>description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>Reference Clock Output</td>
<td></td>
</tr>
<tr>
<td>110</td>
<td>SOSC</td>
<td></td>
</tr>
<tr>
<td>101</td>
<td>MFINTOSC/16 (32 kHz)</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>MFINTOSC (500 kHz)</td>
<td></td>
</tr>
<tr>
<td>011</td>
<td>LFINTOSC</td>
<td></td>
</tr>
<tr>
<td>010</td>
<td>HFINTOSC 16 MHz</td>
<td></td>
</tr>
<tr>
<td>001</td>
<td>Fosc</td>
<td></td>
</tr>
<tr>
<td>000</td>
<td>Fosc/4</td>
<td></td>
</tr>
</tbody>
</table>

### 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
## REGISTER 25-5: SMT1WIN: SMT1 WINDOW INPUT SELECT REGISTER

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Unimplemented</td>
<td>-</td>
</tr>
<tr>
<td>6-5</td>
<td>Unimplemented</td>
<td>-</td>
</tr>
<tr>
<td>4-0</td>
<td>WSEL&lt;4:0&gt;</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Bit 7</td>
<td>U-0</td>
</tr>
<tr>
<td>6</td>
<td>Bit 6</td>
<td>U-0</td>
</tr>
<tr>
<td>5</td>
<td>Bit 5</td>
<td>U-0</td>
</tr>
<tr>
<td>4</td>
<td>Bit 4</td>
<td>R/W-0/0</td>
</tr>
<tr>
<td>3</td>
<td>Bit 3</td>
<td>R/W-0/0</td>
</tr>
<tr>
<td>2</td>
<td>Bit 2</td>
<td>R/W-0/0</td>
</tr>
<tr>
<td>1</td>
<td>Bit 1</td>
<td>R/W-0/0</td>
</tr>
<tr>
<td>0</td>
<td>Bit 0</td>
<td>R/W-0/0</td>
</tr>
</tbody>
</table>

**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**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>11111</td>
<td>Reserved</td>
</tr>
<tr>
<td>11101</td>
<td>CLC4_out</td>
</tr>
<tr>
<td>11100</td>
<td>CLC3_out</td>
</tr>
<tr>
<td>11011</td>
<td>CLC2_out</td>
</tr>
<tr>
<td>11010</td>
<td>CLC1_out</td>
</tr>
<tr>
<td>11001</td>
<td>ZCD1_out</td>
</tr>
<tr>
<td>11000</td>
<td>Reserved</td>
</tr>
<tr>
<td>10111</td>
<td>CMP2_out</td>
</tr>
<tr>
<td>10110</td>
<td>CMP1_out</td>
</tr>
<tr>
<td>10101</td>
<td>NCO1_out</td>
</tr>
<tr>
<td>10100</td>
<td>Reserved</td>
</tr>
<tr>
<td>10011</td>
<td>PWM8_out</td>
</tr>
<tr>
<td>10010</td>
<td>PWM7_out</td>
</tr>
<tr>
<td>10001</td>
<td>PWM6_out</td>
</tr>
<tr>
<td>10000</td>
<td>PWM5_out</td>
</tr>
<tr>
<td>01111</td>
<td>CCP4_out</td>
</tr>
<tr>
<td>01110</td>
<td>CCP3_out</td>
</tr>
<tr>
<td>01101</td>
<td>CCP2_out</td>
</tr>
<tr>
<td>01100</td>
<td>CCP1_out</td>
</tr>
<tr>
<td>01011</td>
<td>TMR6_postscaled</td>
</tr>
<tr>
<td>01010</td>
<td>TMR5_postscaled</td>
</tr>
<tr>
<td>01001</td>
<td>TMR4_postscaled</td>
</tr>
<tr>
<td>01000</td>
<td>TMR3_postscaled</td>
</tr>
<tr>
<td>00111</td>
<td>CLKREF</td>
</tr>
<tr>
<td>00110</td>
<td>SOSC</td>
</tr>
<tr>
<td>00101</td>
<td>MFINTOSC/16 (32 kHz)</td>
</tr>
<tr>
<td>00100</td>
<td>LFINTOSC</td>
</tr>
<tr>
<td>00011</td>
<td>SMTxWINPPS</td>
</tr>
</tbody>
</table>
## REGISTER 25-6: SMT1Sig: SMT1 SIGNAL INPUT SELECT REGISTER

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
</tr>
<tr>
<td>—</td>
<td>—</td>
<td>—</td>
<td>SSEL&lt;4:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 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>: SMT1 Signal Selection bits**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>11111</td>
<td>Reserved</td>
</tr>
<tr>
<td>11101</td>
<td>Reserved</td>
</tr>
<tr>
<td>11100</td>
<td>CLC4_out</td>
</tr>
<tr>
<td>11011</td>
<td>CLC2_out</td>
</tr>
<tr>
<td>11010</td>
<td>CLC1_out</td>
</tr>
<tr>
<td>11001</td>
<td>ZCD1_out</td>
</tr>
<tr>
<td>11000</td>
<td>CMP2_out</td>
</tr>
<tr>
<td>10111</td>
<td>CMP1_out</td>
</tr>
<tr>
<td>10110</td>
<td>NCO1_out</td>
</tr>
<tr>
<td>10101</td>
<td>Reserved</td>
</tr>
<tr>
<td>10100</td>
<td>Reserved</td>
</tr>
<tr>
<td>01111</td>
<td>PWM8_out</td>
</tr>
<tr>
<td>01110</td>
<td>PWM7_out</td>
</tr>
<tr>
<td>01101</td>
<td>PWM6_out</td>
</tr>
<tr>
<td>01100</td>
<td>PWM5_out</td>
</tr>
<tr>
<td>01011</td>
<td>CCP4_out</td>
</tr>
<tr>
<td>01010</td>
<td>CCP3_out</td>
</tr>
<tr>
<td>01001</td>
<td>CCP2_out</td>
</tr>
<tr>
<td>01000</td>
<td>CCP1_out</td>
</tr>
<tr>
<td>00111</td>
<td>TMR6_postscaled</td>
</tr>
<tr>
<td>00110</td>
<td>TMR5_postscaled</td>
</tr>
<tr>
<td>00101</td>
<td>TMR4_postscaled</td>
</tr>
<tr>
<td>00100</td>
<td>TMR3_postscaled</td>
</tr>
<tr>
<td>00011</td>
<td>TMR2_postscaled</td>
</tr>
<tr>
<td>00010</td>
<td>TMR1_postscaled</td>
</tr>
<tr>
<td>00001</td>
<td>TMR0_overflow</td>
</tr>
<tr>
<td>00000</td>
<td>SMTxSIGPPS</td>
</tr>
</tbody>
</table>
**REGISTER 25-7:  SMT1TMRL: SMT TIMER REGISTER – LOW BYTE**

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
</tr>
</tbody>
</table>

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

**bit 7**  **SMT1TMR<7:0>:** Significant bits of the SMT Counter – Low Byte

---

**REGISTER 25-8:  SMT1TMRH: SMT TIMER REGISTER – HIGH BYTE**

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
</tr>
</tbody>
</table>

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

**bit 7**  **SMT1TMR<15:8>:** Significant bits of the SMT Counter – High Byte

---

**REGISTER 25-9:  SMT1TMRU: SMT TIMER REGISTER – UPPER BYTE**

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
</tr>
</tbody>
</table>

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

**bit 7**  **SMT1TMR<23:16>:** Significant bits of the SMT Counter – Upper Byte
REGISTER 25-10: SMT1CPRL: SMT CAPTURED PERIOD REGISTER – LOW BYTE

<table>
<thead>
<tr>
<th>R-x/x</th>
<th>R-x/x</th>
<th>R-x/x</th>
<th>R-x/x</th>
<th>R-x/x</th>
<th>R-x/x</th>
<th>R-x/x</th>
<th>R-x/x</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 SMT1CPR<7:0>: Significant bits of the SMT Period Latch – Low Byte

REGISTER 25-11: SMT1CPRH: SMT CAPTURED PERIOD REGISTER – HIGH BYTE

<table>
<thead>
<tr>
<th>R-x/x</th>
<th>R-x/x</th>
<th>R-x/x</th>
<th>R-x/x</th>
<th>R-x/x</th>
<th>R-x/x</th>
<th>R-x/x</th>
<th>R-x/x</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 SMT1CPR<15:8>: Significant bits of the SMT Period Latch – High Byte

REGISTER 25-12: SMT1CPRU: SMT CAPTURED PERIOD REGISTER – UPPER BYTE

<table>
<thead>
<tr>
<th>R-x/x</th>
<th>R-x/x</th>
<th>R-x/x</th>
<th>R-x/x</th>
<th>R-x/x</th>
<th>R-x/x</th>
<th>R-x/x</th>
<th>R-x/x</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 SMT1CPR<23:16>: Significant bits of the SMT Period Latch – Upper Byte
### REGISTER 25-13: SMT1CPWL: SMT CAPTURED PULSE WIDTH REGISTER – LOW BYTE

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>SMT1CPW&lt;7:0&gt;</th>
</tr>
</thead>
</table>

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

**SMT1CPW<7:0>:** Significant bits of the SMT PW Latch – Low Byte

### REGISTER 25-14: SMT1CPWH: SMT CAPTURED PULSE WIDTH REGISTER – HIGH BYTE

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>SMT1CPW&lt;15:8&gt;</th>
</tr>
</thead>
</table>

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

**SMT1CPW<15:8>:** Significant bits of the SMT PW Latch – High Byte

### REGISTER 25-15: SMT1CPWU: SMT CAPTURED PULSE WIDTH REGISTER – UPPER BYTE

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>SMT1CPW&lt;23:16&gt;</th>
</tr>
</thead>
</table>

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

**SMT1CPW<23:16>:** Significant bits of the SMT PW Latch – Upper Byte
REGISTER 25-16: SMT1PR: SMT PERIOD REGISTER – LOW BYTE

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 6</th>
<th>bit 5</th>
<th>bit 4</th>
<th>bit 3</th>
<th>bit 2</th>
<th>bit 1</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>R/W-x/1</td>
<td>R/W-x/1</td>
<td>R/W-x/1</td>
<td>R/W-x/1</td>
<td>R/W-x/1</td>
<td>R/W-x/1</td>
<td>R/W-x/1</td>
<td>R/W-x/1</td>
</tr>
</tbody>
</table>

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  SMT1PR<7:0>: Significant bits of the SMT Timer Value for Period Match – Low Byte

REGISTER 25-17: SMT1PRH: SMT PERIOD REGISTER – HIGH BYTE

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 6</th>
<th>bit 5</th>
<th>bit 4</th>
<th>bit 3</th>
<th>bit 2</th>
<th>bit 1</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>R/W-x/1</td>
<td>R/W-x/1</td>
<td>R/W-x/1</td>
<td>R/W-x/1</td>
<td>R/W-x/1</td>
<td>R/W-x/1</td>
<td>R/W-x/1</td>
<td>R/W-x/1</td>
</tr>
</tbody>
</table>

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  SMT1PR<15:8>: Significant bits of the SMT Timer Value for Period Match – High Byte

REGISTER 25-18: SMT1PRU: SMT PERIOD REGISTER – UPPER BYTE

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 6</th>
<th>bit 5</th>
<th>bit 4</th>
<th>bit 3</th>
<th>bit 2</th>
<th>bit 1</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>R/W-x/1</td>
<td>R/W-x/1</td>
<td>R/W-x/1</td>
<td>R/W-x/1</td>
<td>R/W-x/1</td>
<td>R/W-x/1</td>
<td>R/W-x/1</td>
<td>R/W-x/1</td>
</tr>
</tbody>
</table>

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  SMT1PR<23:16>: Significant bits of the SMT Timer Value for Period Match – Upper Byte
### TABLE 25-3: SUMMARY OF REGISTERS ASSOCIATED WITH SMT1

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMT1CON0</td>
<td>EN</td>
<td>—</td>
<td>STP</td>
<td>WPOL</td>
<td>—</td>
<td>SPOL</td>
<td>—</td>
<td>—</td>
<td>SMT1PS&lt;1:0&gt;</td>
</tr>
<tr>
<td>SMT1CON1</td>
<td>GO</td>
<td>REPEAT</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>SMT1STAT</td>
<td>CPRUP</td>
<td>CPWUP</td>
<td>RST</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>SMT1CLK</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CSEL&lt;2:0&gt;</td>
</tr>
<tr>
<td>SMT1SIG</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>SMT1WIN</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>SMT1TMRL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>WSEL&lt;4:0&gt;</td>
</tr>
<tr>
<td>SMT1TMRL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>TMR&lt;7:0&gt;</td>
</tr>
<tr>
<td>SMT1TMRL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>TMR&lt;15:8&gt;</td>
</tr>
<tr>
<td>SMT1TMRL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>TMR&lt;23:16&gt;</td>
</tr>
<tr>
<td>SMT1CPRL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CPR&lt;7:0&gt;</td>
</tr>
<tr>
<td>SMT1CPRL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CPR&lt;15:8&gt;</td>
</tr>
<tr>
<td>SMT1CPRL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CPR&lt;23:16&gt;</td>
</tr>
<tr>
<td>SMT1CPRL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CPW&lt;7:0&gt;</td>
</tr>
<tr>
<td>SMT1CPRL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CPW&lt;15:8&gt;</td>
</tr>
<tr>
<td>SMT1CPRL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CPW&lt;23:16&gt;</td>
</tr>
<tr>
<td>SMT1PRL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>PR&lt;7:0&gt;</td>
</tr>
<tr>
<td>SMT1PRL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>PR&lt;15:8&gt;</td>
</tr>
<tr>
<td>SMT1PRL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>PR&lt;23:16&gt;</td>
</tr>
</tbody>
</table>

**Legend:** — = unimplemented read as ‘0’. Shaded cells are not used for SMT1 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)F2X/4X/5XK42 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 26-1: SIMPLIFIED CWG BLOCK DIAGRAM (HALF-BRIDGE MODE, MODE<2:0> = 100)
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)
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)

MODE<2:0> = 010: Forward
MODE<2:0> = 011: Reverse

- CWG Clock
  - Rising Dead-Band Block
  - Falling Dead-Band Block

- CWG Data Input
  - Auto-shutdown source (CWGxAS1 register)
  - REN
  - SHUTDOWN = 0

- CWG Data
  - CWG Clock
  - CWG Data A
    - POLA
    - CWG1A
  - CWG Data B
    - POLB
    - CWG1B
  - CWG Data C
    - POLC
    - CWG1C
  - CWG Data D
    - POLD
    - CWG1D

- SHUTDOWN:
  - FREEZE

- Modes:
  - MODE<2:0> = 010: Forward
  - MODE<2:0> = 011: Reverse

- Signals:
  - '1' and '0'
  - High-Z

- Logic Gates:
  - XOR
  - AND
  - OR

- Clock Signals:
  - CWG Clock
  - CWG Data

- Signal In and Signal Out

- Modes and Signals Interaction:
  - 101: Forward
  - 110: Reverse

- Auto-shutdown source (CWGxAS1 register)
In Forward Full-Bridge mode (MODE<2:0> = 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 (MODE<2:0> = 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 MODE<0> bit of the CWGxCON0 while keeping MODE<2:1> static, without disabling the CWG module.

FIGURE 26-7: EXAMPLE OF FULL-BRIDGE OUTPUT

<table>
<thead>
<tr>
<th>Forward Mode</th>
<th>Reverse Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWG1A(2)</td>
<td>CWG1A(2)</td>
</tr>
<tr>
<td>CWG1B(2)</td>
<td>CWG1B(2)</td>
</tr>
<tr>
<td>CWG1C(2)</td>
<td>CWG1C(2)</td>
</tr>
<tr>
<td>CWG1D(2)</td>
<td>CWG1D(2)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pulse Width</th>
<th>Pulse Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>(1)</td>
</tr>
</tbody>
</table>

Note 1: A rising CWG data input creates a rising event on the modulated output.

2: Output signals shown as active-high; all POLy bits are clear.
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)

MODE<2:0> = 000: Asynchronous
MODE<2:0> = 001: Synchronous
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>
<thead>
<tr>
<th>Source Peripheral</th>
<th>Signal Name</th>
<th>ISM&lt;2:0&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWGxPPS</td>
<td>Pin selected by CWGxPPS</td>
<td>000</td>
</tr>
<tr>
<td>CCP1</td>
<td>CCP1 Output</td>
<td>001</td>
</tr>
<tr>
<td>CCP2</td>
<td>CCP2 Output</td>
<td>010</td>
</tr>
<tr>
<td>PWM3</td>
<td>PWM3 Output</td>
<td>011</td>
</tr>
<tr>
<td>PWM4</td>
<td>PWM4 Output</td>
<td>100</td>
</tr>
<tr>
<td>CMP1</td>
<td>Comparator 1 Output</td>
<td>101</td>
</tr>
<tr>
<td>CMP2</td>
<td>Comparator 2 Output</td>
<td>110</td>
</tr>
<tr>
<td>DSM</td>
<td>Data signal modulator output</td>
<td>111</td>
</tr>
</tbody>
</table>

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 RxypSS 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**

\[
\begin{align*}
T_{DEAD-BAND\ MIN} &= \frac{1}{F_{CWG\ CLOCK}} \times DBx <4:0> \\
T_{DEAD-BAND\ MAX} &= \frac{1}{F_{CWG\ CLOCK}} \times DBx <4:0> + 1 \\
T_{JITTER} &= T_{DEAD-BAND\ MAX} - T_{DEAD-BAND\ MIN} \\
T_{JITTER} &= \frac{1}{F_{CWG\ CLOCK}} \\
T_{DEAD-BAND\ MAX} &= \frac{T_{DEAD-BAND\ MIN} + T_{JITTER}}{2}
\end{align*}
\]

**EXAMPLE**

\[
\begin{align*}
DBx<4:0> &= 000.4 = 10 \\
F_{CWG\ CLOCK} &= 8\ MHz \\
T_{JITTER} &= \frac{1}{8\ MHz} = 125\ ns \\
T_{DEAD-BAND\ MIN} &= 125\ ns \times 10 = 125\ \mu s \\
T_{DEAD-BAND\ MAX} &= 1.25\ \mu s + 0.125\ \mu s = 1.37\ \mu s
\end{align*}
\]
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 LSBD<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 LSBD<1:0> and LSAC<1:0> bits of the CWGxAS0 register (Register 26-6). The LSBD<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 LSBD<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, LSBD = 01)
FIGURE 26-16: SHUTDOWN FUNCTIONALITY, AUTO-RESTART ENABLED (REN = 1, LSAC = 01, LSBD = 01)

- CWG Input Source
- Shutdown Source
- SHUTDOWN
- CWGxA
- CWGxC
- CWGxB
- CWGxD

- No Shutdown
- Tri-State (No Pulse)
- Tri-State (No Pulse)

- Shutdown Event Ceases
- REN auto-cleared by hardware

- Output Resumes
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.

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>Bit Name Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWG1</td>
<td>CWG1</td>
</tr>
<tr>
<td>CWG2</td>
<td>CWG2</td>
</tr>
<tr>
<td>CWG3</td>
<td>CWG3</td>
</tr>
</tbody>
</table>

REGISTER 26-1: CWGxCON0: CWG CONTROL REGISTER 0

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>EN: CWGx Enable bit</td>
<td></td>
<td>1 = Module is enabled, 0 = Module is disabled</td>
</tr>
<tr>
<td>6</td>
<td>LD: CWGx Load Buffers bit&lt;sup&gt;(1)&lt;/sup&gt;</td>
<td></td>
<td>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</td>
</tr>
<tr>
<td>5-3</td>
<td>Unimplemented</td>
<td></td>
<td>Read as ‘0’</td>
</tr>
<tr>
<td>2-0</td>
<td>MODE&lt;2:0&gt;: CWGx Mode bits</td>
<td></td>
<td>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</td>
</tr>
</tbody>
</table>

Note 1: This bit can only be set after EN = 1; it cannot be set in the same cycle when EN is set.

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
## REGISTER 26-2: CWGxCON1: CWG CONTROL REGISTER 1

<table>
<thead>
<tr>
<th>bit 7-6</th>
<th>bit 5</th>
<th>bit 4</th>
<th>bit 3</th>
<th>bit 2</th>
<th>bit 1</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unimplemented</td>
<td>IN: CWG Input Value bit (read-only)</td>
<td>Unimplemented</td>
<td>POLD: CWGxD Output Polarity bit</td>
<td>POLC: CWGxC Output Polarity bit</td>
<td>POLB: CWGxB Output Polarity bit</td>
<td>POLA: CWGxA Output Polarity bit</td>
</tr>
</tbody>
</table>

### 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

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 6</th>
<th>bit 5</th>
<th>bit 4</th>
<th>bit 3</th>
<th>bit 2</th>
<th>bit 1</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>—</td>
<td>IN</td>
<td>—</td>
<td>POLD</td>
<td>POLC</td>
<td>POLB</td>
<td>POLA</td>
</tr>
</tbody>
</table>

- ** bit 7 **
  - **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
## REGISTER 26-3: CWGxCLK: CWGx CLOCK INPUT SELECTION REGISTER

| bit 7-1 | Unimplemented: Read as ’0’ |
| bit 0 | CS: CWG Clock Source Selection bits |

<table>
<thead>
<tr>
<th>CS</th>
<th>CWG1</th>
<th>CWG2</th>
<th>CWG3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>HFINTOSC (1)</td>
<td>HFINTOSC (1)</td>
<td>HFINTOSC (1)</td>
</tr>
<tr>
<td>0</td>
<td>Fosc</td>
<td>Fosc</td>
<td>Fosc</td>
</tr>
</tbody>
</table>

**Note 1:** HFINTOSC remains operating during Sleep.

**Legend:**
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as ‘0’
- **u** = Bit is unchanged
- **x** = Bit is unknown
- **’1’** = Bit is set
- **’0’** = Bit is cleared
- **q** = Value depends on condition
- **-n/n** = Value at POR and BOR/Value at all other Resets

<sup>(1)</sup> HFINTOSC remains operating during Sleep.
REGISTER 26-4: CWGxISM: CWGx INPUT SELECTION REGISTER

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ISM&lt;4:0&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 ISM<4:0>: CWG Data Input Selection Multiplexer Select bits

<table>
<thead>
<tr>
<th>ISM&lt;4:0&gt;</th>
<th>CWG1 Input Selection</th>
<th>CWG2 Input Selection</th>
<th>CWG3 Input Selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>11111-10011</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>10010</td>
<td>CLC4_out</td>
<td>CLC4_out</td>
<td>CLC4_out</td>
</tr>
<tr>
<td>10001</td>
<td>CLC3_out</td>
<td>CLC3_out</td>
<td>CLC3_out</td>
</tr>
<tr>
<td>10000</td>
<td>CLC2_out</td>
<td>CLC2_out</td>
<td>CLC2_out</td>
</tr>
<tr>
<td>01111</td>
<td>CLC1_out</td>
<td>CLC1_out</td>
<td>CLC1_out</td>
</tr>
<tr>
<td>01110</td>
<td>DSM_out</td>
<td>DSM_out</td>
<td>DSM_out</td>
</tr>
<tr>
<td>01101</td>
<td>CMP2OUT</td>
<td>CMP2OUT</td>
<td>CMP2OUT</td>
</tr>
<tr>
<td>01100</td>
<td>CMP1OUT</td>
<td>CMP1OUT</td>
<td>CMP1OUT</td>
</tr>
<tr>
<td>01011</td>
<td>NCO1OUT</td>
<td>NCO1OUT</td>
<td>NCO1OUT</td>
</tr>
<tr>
<td>01010-01001</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>01000</td>
<td>PWM8OUT</td>
<td>PWM8OUT</td>
<td>PWM8OUT</td>
</tr>
<tr>
<td>00111</td>
<td>PWM7OUT</td>
<td>PWM7OUT</td>
<td>PWM7OUT</td>
</tr>
<tr>
<td>00110</td>
<td>PWM6OUT</td>
<td>PWM6OUT</td>
<td>PWM6OUT</td>
</tr>
<tr>
<td>00101</td>
<td>PWM5OUT</td>
<td>PWM5OUT</td>
<td>PWM5OUT</td>
</tr>
<tr>
<td>00100</td>
<td>CCP4_out</td>
<td>CCP4_out</td>
<td>CCP4_out</td>
</tr>
<tr>
<td>00011</td>
<td>CCP3_out</td>
<td>CCP3_out</td>
<td>CCP3_out</td>
</tr>
<tr>
<td>00010</td>
<td>CCP2_out</td>
<td>CCP2_out</td>
<td>CCP2_out</td>
</tr>
<tr>
<td>00001</td>
<td>CCP1_out</td>
<td>CCP1_out</td>
<td>CCP1_out</td>
</tr>
<tr>
<td>00000</td>
<td>Pin selected by CWG1PPS</td>
<td>Pin selected by CWG2PPS</td>
<td>Pin selected by CWG3PPS</td>
</tr>
</tbody>
</table>
### REGISTER 26-5: CWGxSTR(1): CWG STEERING CONTROL REGISTER

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 6</th>
<th>bit 5</th>
<th>bit 4</th>
<th>bit 3</th>
<th>bit 2</th>
<th>bit 1</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>OVRD</td>
<td>OVRC</td>
<td>OVRB</td>
<td>OVRA</td>
<td>STRD(2)</td>
<td>STRC(2)</td>
<td>STRB(2)</td>
<td>STRA(2)</td>
</tr>
</tbody>
</table>

#### Legend:
- **R** = Readable bit
- **W** = Writable bit
- **u** = Bit is unchanged
- **x** = Bit is unknown
- **'1'** = Bit is set
- **'0'** = Bit is cleared
- **q** = Value depends on condition
- **-n/n** = Value at POR and BOR/Value at all other Resets

#### Description:
- **OVRD**: Steering Data D bit
  - 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

- **OVRC**: Steering Data C bit
  - 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

- **OVRB**: Steering Data B bit
  - 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

- **OVRA**: Steering Data A bit
  - 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**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
<th>Value at POR and BOR</th>
<th>Value at all other Resets</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>SHUTDOWN: Auto-Shutdown Event Status bit(1,2)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = An auto-shutdown state is in effect</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = No auto-shutdown event has occurred</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>REN: Auto-Restart Enable bit</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = Auto-restart is enabled</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = Auto-restart is disabled</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5-4</td>
<td>LSBD&lt;1:0&gt;: CWGxB and CWGxD Auto-Shutdown State Control bits</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>11 = A logic '1' is placed on CWGxB/D when an auto-shutdown event occurs.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10 = A logic '0' is placed on CWGxB/D when an auto-shutdown event occurs.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>01 = Pin is tri-stated on CWGxB/D when an auto-shutdown event occurs.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>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.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3-2</td>
<td>LSAC&lt;1:0&gt;: CWGxA and CWGxC Auto-Shutdown State Control bits</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>11 = A logic '1' is placed on CWGxA/C when an auto-shutdown event occurs.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10 = A logic '0' is placed on CWGxA/C when an auto-shutdown event occurs.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>01 = Pin is tri-stated on CWGxA/C when an auto-shutdown event occurs.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>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.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-0</td>
<td>Unimplemented: Read as '0'</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

**Note 1:** This bit may be written while EN = 0 (Register 26-1), to place the outputs into the shutdown configuration.

**Note 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

<table>
<thead>
<tr>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>AS6E</td>
<td>AS5E</td>
<td>AS4E</td>
<td>AS3E</td>
<td>AS2E</td>
<td>AS1E</td>
<td>AS0E</td>
</tr>
</tbody>
</table>

| bit 7 | Unimplemented Read as ‘0’ |
| bit 6 | **AS6E**: CWG Auto-shutdown Source 6 Enable bit |
|      | 1 = Auto-shutdown for Source 6 is enabled |
|      | 0 = Auto-shutdown for Source 6 is disabled |
|      | **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 |

**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
REGISTER 26-8:  CWGxDBR: CWG RISING DEAD-BAND COUNT REGISTER

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWGxCON0</td>
<td>EN</td>
<td>LD</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MODE&lt;2:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>424</td>
</tr>
<tr>
<td>CWGxCON1</td>
<td>—</td>
<td>—</td>
<td>IN</td>
<td>—</td>
<td>—</td>
<td>POLD</td>
<td>POLC</td>
<td>POLB</td>
<td>POLA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>425</td>
</tr>
<tr>
<td>CWGxCLK</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>426</td>
</tr>
<tr>
<td>CWGxISM</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CS&lt;4:0&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>427</td>
</tr>
<tr>
<td>CWGxSTR</td>
<td>OVRD</td>
<td>OVRC</td>
<td>OVRB</td>
<td>OVRA</td>
<td>STRD</td>
<td>STRC</td>
<td>STRB</td>
<td>STRA</td>
<td>OVRD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>428</td>
</tr>
<tr>
<td>CWGxAS0</td>
<td>SHUTDOWN</td>
<td>REN</td>
<td>LSBD&lt;1:0&gt;</td>
<td>LSAC&lt;1:0&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>AS&lt;5:0&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>429</td>
</tr>
<tr>
<td>CWGxAS1</td>
<td>—</td>
<td>AS6E</td>
<td>AS5E</td>
<td>AS4E</td>
<td>AS3E</td>
<td>AS2E</td>
<td>AS1E</td>
<td>AS0E</td>
<td>AS&lt;5:0&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>430</td>
</tr>
<tr>
<td>CWGxDBR</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>DBR&lt;5:0&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>431</td>
</tr>
<tr>
<td>CWGxDBF</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>DBF&lt;5:0&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>431</td>
</tr>
</tbody>
</table>

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
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 runtime 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 CLC 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>.

---

**Note:** Data selections are undefined at power-up.
## TABLE 27-1: CLCx DATA INPUT SELECTION

<table>
<thead>
<tr>
<th>Dy5&lt;5:0&gt; Value</th>
<th>CLCx Input Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>111111 [63]</td>
<td>Reserved</td>
</tr>
<tr>
<td>.</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td></td>
</tr>
<tr>
<td>110100 [52]</td>
<td>Reserved</td>
</tr>
<tr>
<td>110011 [51]</td>
<td>CWG3B_out</td>
</tr>
<tr>
<td>110010 [50]</td>
<td>CWG3A_out</td>
</tr>
<tr>
<td>110001 [49]</td>
<td>CWG2B_out</td>
</tr>
<tr>
<td>110000 [48]</td>
<td>CWG2A_out</td>
</tr>
<tr>
<td>101111 [47]</td>
<td>CWG1B_out</td>
</tr>
<tr>
<td>101110 [46]</td>
<td>CWG1A_out</td>
</tr>
<tr>
<td>101101 [45]</td>
<td>SS1</td>
</tr>
<tr>
<td>101100 [44]</td>
<td>SCK1</td>
</tr>
<tr>
<td>101011 [43]</td>
<td>SDO1</td>
</tr>
<tr>
<td>101010 [42]</td>
<td>Reserved</td>
</tr>
<tr>
<td>101001 [41]</td>
<td>UART2_tx_out</td>
</tr>
<tr>
<td>101000 [40]</td>
<td>UART1_tx_out</td>
</tr>
<tr>
<td>100111 [39]</td>
<td>CLC4_out</td>
</tr>
<tr>
<td>100110 [38]</td>
<td>CLC3_out</td>
</tr>
<tr>
<td>100101 [37]</td>
<td>CLC2_out</td>
</tr>
<tr>
<td>100100 [36]</td>
<td>CLC1_out</td>
</tr>
<tr>
<td>100011 [35]</td>
<td>DSM1_out</td>
</tr>
<tr>
<td>100010 [34]</td>
<td>IOC_flag</td>
</tr>
<tr>
<td>100001 [33]</td>
<td>ZCD_out</td>
</tr>
<tr>
<td>100000 [32]</td>
<td>CMP2_out</td>
</tr>
<tr>
<td>011111 [31]</td>
<td>CMP1_out</td>
</tr>
<tr>
<td>011110 [30]</td>
<td>NCO1_out</td>
</tr>
<tr>
<td>011101 [29]</td>
<td>Reserved</td>
</tr>
<tr>
<td>011100 [28]</td>
<td>Reserved</td>
</tr>
<tr>
<td>011011 [27]</td>
<td>PWM8_out</td>
</tr>
<tr>
<td>011010 [26]</td>
<td>PWM7_out</td>
</tr>
<tr>
<td>011001 [25]</td>
<td>PWM6_out</td>
</tr>
<tr>
<td>011000 [24]</td>
<td>PWM5_out</td>
</tr>
<tr>
<td>010111 [23]</td>
<td>CCP4_out</td>
</tr>
<tr>
<td>010110 [22]</td>
<td>CCP3_out</td>
</tr>
<tr>
<td>010101 [21]</td>
<td>CCP2_out</td>
</tr>
<tr>
<td>010100 [20]</td>
<td>CCP1_out</td>
</tr>
<tr>
<td>010011 [19]</td>
<td>SMT1_out</td>
</tr>
<tr>
<td>010010 [18]</td>
<td>TMR6_out</td>
</tr>
<tr>
<td>010001 [17]</td>
<td>TMR5_overflow</td>
</tr>
<tr>
<td>010000 [16]</td>
<td>TMR4_out</td>
</tr>
<tr>
<td>001111 [15]</td>
<td>TMR3_overflow</td>
</tr>
</tbody>
</table>

### TABLE 27-1: CLCx DATA INPUT SELECTION (CONTINUED)

<table>
<thead>
<tr>
<th>Dy5&lt;5:0&gt; Value</th>
<th>CLCx Input Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>001110 [14]</td>
<td>TMR2_out</td>
</tr>
<tr>
<td>001101 [13]</td>
<td>TMR1_overflow</td>
</tr>
<tr>
<td>001100 [12]</td>
<td>TMR0_overflow</td>
</tr>
<tr>
<td>001010 [10]</td>
<td>ADCRC</td>
</tr>
<tr>
<td>001001 [9]</td>
<td>SOIC</td>
</tr>
<tr>
<td>001000 [8]</td>
<td>MFINTOSC (32 kHz)</td>
</tr>
<tr>
<td>000111 [7]</td>
<td>MFINTOSC (500 kHz)</td>
</tr>
<tr>
<td>000110 [6]</td>
<td>LFINTOSC</td>
</tr>
<tr>
<td>000101 [5]</td>
<td>HFINTOSC</td>
</tr>
<tr>
<td>000100 [4]</td>
<td>Fosc</td>
</tr>
<tr>
<td>000011 [3]</td>
<td>CLCIN3PPS</td>
</tr>
<tr>
<td>000010 [2]</td>
<td>CLCIN2PPS</td>
</tr>
<tr>
<td>000001 [1]</td>
<td>CLCIN1PPS</td>
</tr>
<tr>
<td>000000 [0]</td>
<td>CLCIN0PPS</td>
</tr>
</tbody>
</table>
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 or 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

<table>
<thead>
<tr>
<th>CLCxGLSy</th>
<th>GyPOL</th>
<th>Gate Logic</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x55</td>
<td>1</td>
<td>AND</td>
</tr>
<tr>
<td>0x55</td>
<td>0</td>
<td>NAND</td>
</tr>
<tr>
<td>0xAA</td>
<td>1</td>
<td>NOR</td>
</tr>
<tr>
<td>0xAA</td>
<td>0</td>
<td>OR</td>
</tr>
<tr>
<td>0x00</td>
<td>0</td>
<td>Logic 0</td>
</tr>
<tr>
<td>0x00</td>
<td>1</td>
<td>Logic 1</td>
</tr>
</tbody>
</table>

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.

**Note:** Data gating is undefined at power-up.
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

Note: All controls are undefined at power-up.
FIGURE 27-3: PROGRAMMABLE LOGIC FUNCTIONS

AND-OR

MODE<2:0> = 000

OR-XOR

MODE<2:0> = 001

4-input AND

MODE<2:0> = 010

S-R Latch

MODE<2:0> = 011

1-Input D Flip-Flop with S and R

MODE<2:0> = 100

2-Input D Flip-Flop with R

MODE<2:0> = 101

J-K Flip-Flop with R

MODE<2:0> = 110

1-Input Transparent Latch with S and R

MODE<2:0> = 111
27.7  Register Definitions: CLC Control

REGISTER 27-1:  CLCxCON: CONFIGURABLE LOGIC CELL CONTROL REGISTER

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>U-0</th>
<th>R-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN</td>
<td>—</td>
<td>OUT</td>
<td>INTP</td>
<td>INTN</td>
<td>MODE&lt;2:0&gt;</td>
<td>bit 7</td>
<td>bit 0</td>
</tr>
</tbody>
</table>

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 LCPO; 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

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6-4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>POL</td>
<td>Unimplemented</td>
<td>G4POL</td>
<td>G3POL</td>
<td>G2POL</td>
<td>G1POL</td>
</tr>
</tbody>
</table>

#### 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

<table>
<thead>
<tr>
<th>bit 7-6</th>
<th>bit 5-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unimplemented</td>
<td>D1S&lt;5:0&gt;</td>
</tr>
</tbody>
</table>

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'
bite 5-0   D1S<5:0>: CLCx Data 1 Input Selection bits
See Table 27-1.

REGISTER 27-4: CLCxSEL1: GENERIC CLCx DATA 1 SELECT REGISTER

<table>
<thead>
<tr>
<th>bit 7-6</th>
<th>bit 5-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unimplemented</td>
<td>D2S&lt;5:0&gt;</td>
</tr>
</tbody>
</table>

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'
bite 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

<table>
<thead>
<tr>
<th>bit 7-6</th>
<th>bit 5-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unimplemented</td>
<td>D3S&lt;5:0&gt;</td>
</tr>
</tbody>
</table>

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'
bite 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

<table>
<thead>
<tr>
<th>bit 7-6</th>
<th>bit 5-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unimplemented</td>
<td>D4S&lt;5:0&gt;</td>
</tr>
</tbody>
</table>

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'
bite 5-0   D4S<5:0>: CLCx Data 4 Input Selection bits
See Table 27-1.
## REGISTER 27-7:  CLCxGLS0: GATE 0 LOGIC SELECT REGISTER

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 6</th>
<th>bit 5</th>
<th>bit 4</th>
<th>bit 3</th>
<th>bit 2</th>
<th>bit 1</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
</tr>
<tr>
<td>G1D4T</td>
<td>G1D4N</td>
<td>G1D3T</td>
<td>G1D3N</td>
<td>G1D2T</td>
<td>G1D2N</td>
<td>G1D1T</td>
<td>G1D1N</td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
</tr>
</thead>
<tbody>
<tr>
<td>G2D4T</td>
<td>G2D4N</td>
<td>G2D3T</td>
<td>G2D3N</td>
<td>G2D2T</td>
<td>G2D2N</td>
<td>G2D1T</td>
<td>G2D1N</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 0</th>
</tr>
</thead>
</table>

**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

- **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

- **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

- **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

- **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

- **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

- **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

- **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

- **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

<table>
<thead>
<tr>
<th>Bit</th>
<th>R/W-x/u</th>
<th>G3D4T</th>
<th>G3D4N</th>
<th>G3D3T</th>
<th>G3D3N</th>
<th>G3D2T</th>
<th>G3D2N</th>
<th>G3D1T</th>
<th>G3D1N</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>R-x/u</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>R-x/u</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>R-x/u</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>R-x/u</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>R-x/u</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>R-x/u</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>R-x/u</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>R-x/u</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

#### 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

<table>
<thead>
<tr>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
</tr>
</thead>
<tbody>
<tr>
<td>G4D4T</td>
<td>G4D4N</td>
<td>G4D3T</td>
<td>G4D3N</td>
<td>G4D2T</td>
<td>G4D2N</td>
<td>G4D1T</td>
<td>G4D1N</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CLC4OUT</td>
<td>CLC3OUT</td>
<td>CLC2OUT</td>
<td>CLC1OUT</td>
</tr>
</tbody>
</table>

bit 7

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLCxCON</td>
<td>EN</td>
<td>—</td>
<td>OUT</td>
<td>INTP</td>
<td>INTN</td>
<td>MODE&lt;2:0&gt;</td>
<td></td>
<td></td>
<td>440</td>
</tr>
<tr>
<td>CLCxPOL</td>
<td>POL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>G4POL</td>
<td>G3POL</td>
<td>G2POL</td>
<td>G1POL</td>
</tr>
<tr>
<td>CLCxSEL0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>D1S&lt;5:0&gt;</td>
<td></td>
<td>442</td>
</tr>
<tr>
<td>CLCxSEL1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>D2S&lt;5:0&gt;</td>
<td></td>
<td>442</td>
</tr>
<tr>
<td>CLCxSEL2</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>D3S&lt;5:0&gt;</td>
<td></td>
<td>442</td>
</tr>
<tr>
<td>CLCxSEL3</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>D4S&lt;5:0&gt;</td>
<td></td>
<td>442</td>
</tr>
<tr>
<td>CLCxGLS0</td>
<td>G1D4T</td>
<td>G1D4N</td>
<td>G1D3T</td>
<td>G1D3N</td>
<td>G1D2T</td>
<td>G1D2N</td>
<td>G1D1T</td>
<td>G1D1N</td>
<td>443</td>
</tr>
<tr>
<td>CLCxGLS1</td>
<td>G2D4T</td>
<td>G2D4N</td>
<td>G2D3T</td>
<td>G2D3N</td>
<td>G2D2T</td>
<td>G2D2N</td>
<td>G2D1T</td>
<td>G2D1N</td>
<td>444</td>
</tr>
<tr>
<td>CLCxGLS2</td>
<td>G3D4T</td>
<td>G3D4N</td>
<td>G3D3T</td>
<td>G3D3N</td>
<td>G3D2T</td>
<td>G3D2N</td>
<td>G3D1T</td>
<td>G3D1N</td>
<td>445</td>
</tr>
<tr>
<td>CLCxGLS3</td>
<td>G4D4T</td>
<td>G4D4N</td>
<td>G4D3T</td>
<td>G4D3N</td>
<td>G4D2T</td>
<td>G4D2N</td>
<td>G4D1T</td>
<td>G4D1N</td>
<td>446</td>
</tr>
<tr>
<td>CLCDATA</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CLC4OUT</td>
<td>CLC3OUT</td>
<td>CLC2OUT</td>
<td>CLC1OUT</td>
</tr>
</tbody>
</table>

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 applications that require 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.
FIGURE 28-1: DIRECT DIGITAL SYNTHESIS MODULE SIMPLIFIED BLOCK DIAGRAM

Note 1: The increment registers are double-buffered to allow for value changes to be made without first disabling the NCO module. The full increment value is loaded into the buffer registers on the second rising edge of the NCOx_clk signal that occurs immediately after a write to NCOxINCL register. The buffers are not user-accessible and are shown here for reference.
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
- NCO1INCU

When the NCO module is enabled, the NCO1INCU 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

- NCOx Clock Source
- NCOx Increment Value
- NCOx Accumulator Value
- NCOx Overflow
- NCOx Interrupt
- NCOx Output FDC Mode
- NCOx Output PF Mode
- NCOxPWS = 000
- NCOx Output PF Mode
- NCOxPWS = 001
28.8 NCO Control Registers

REGISTER 28-1: NCO1CON: NCO CONTROL REGISTER

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>U-0</th>
<th>R-0/0</th>
<th>R/W-0/0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>PFM</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN</td>
<td>—</td>
<td>OUT</td>
<td>POL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>bit 7-5 PWS&lt;2:0&gt;</th>
<th>bit 4</th>
<th>bit 3-0 CKS&lt;3:0&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>111 = NCO1 output is active for 128 input clock periods</td>
<td>Unimplemented: Read as ‘0’</td>
<td>1111 = Reserved</td>
</tr>
<tr>
<td>110 = NCO1 output is active for 64 input clock periods</td>
<td></td>
<td>1110 = CLC4_out</td>
</tr>
<tr>
<td>101 = NCO1 output is active for 32 input clock periods</td>
<td></td>
<td>1001 = CLC3_out</td>
</tr>
<tr>
<td>100 = NCO1 output is active for 16 input clock periods</td>
<td></td>
<td>1000 = CLC2_out</td>
</tr>
<tr>
<td>011 = NCO1 output is active for 8 input clock periods</td>
<td></td>
<td>0111 = CLC1_out</td>
</tr>
<tr>
<td>010 = NCO1 output is active for 4 input clock periods</td>
<td></td>
<td>0110 = CLKREF_out</td>
</tr>
<tr>
<td>001 = NCO1 output is active for 2 input clock periods</td>
<td></td>
<td>0101 = SOSC</td>
</tr>
<tr>
<td>000 = NCO1 output is active for 1 input clock period</td>
<td></td>
<td>0100 = MFINTOSC/4 (32 kHz)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0011 = MFINTOSC (500 kHz)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0010 = LFINTOSC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001 = HFINTOSC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0000 = Fosc</td>
</tr>
</tbody>
</table>

**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

**Note 1:** N1PWS applies only when operating in Pulse Frequency mode.

**Note 2:** If NCO1 pulse width is greater than NCO1 overflow period, operation is undefined.
## REGISTER 28-3: NCOACCL: NCO1 ACCUMULATOR REGISTER – LOW BYTE

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td>ACC&lt;7:0&gt;</td>
<td>bit 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

bit 7-0 \(\text{ACC<7:0>}:\) NCO1 Accumulator, Low Byte

## REGISTER 28-4: NCOACCH: NCO1 ACCUMULATOR REGISTER – HIGH BYTE

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td>ACC&lt;15:8&gt;</td>
<td>bit 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

bit 7-0 \(\text{ACC<15:8>}:\) NCO1 Accumulator, High Byte
REGISTER 28-5:  NCO1ACC: NCO1 ACCUMULATOR REGISTER – UPPER BYTE(1)

<table>
<thead>
<tr>
<th>bit 7-4</th>
<th>bit 3-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unimplemented:</td>
<td>Read as ‘0’</td>
</tr>
</tbody>
</table>

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(1,2)

<table>
<thead>
<tr>
<th>bit 7-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>INC&lt;7:0&gt;:</td>
</tr>
</tbody>
</table>

Note 1: The logical increment spans NCO1INCU:NCO1INCH:NCO1INCL.
2: NCO1INC is double-buffered as INCBUF; INCBUF is updated on the next falling edge of NCOCLK after writing to NCO1INCL; NCO1INCU and NCO1INCH should be written prior to writing NCO1INCL.

REGISTER 28-7:  NCO1INCH: NCO1 INCREMENT REGISTER – HIGH BYTE(1)

<table>
<thead>
<tr>
<th>bit 7-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>INC&lt;15:8&gt;:</td>
</tr>
</tbody>
</table>

Note 1: The logical increment spans NCO1INCU:NCO1INCH:NCO1INCL.
REGISTER 28-8: NCO1INCU: NCO1 INCREMENT REGISTER – UPPER BYTE(1)

<table>
<thead>
<tr>
<th></th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>INC&lt;19:16&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 NCO1INCU:NCO1INCH:NCO1INCL.

TABLE 28-1: SUMMARY OF REGISTERS ASSOCIATED WITH NCO

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>NCO1CON</td>
<td>N1EN</td>
<td>—</td>
<td>N1OUT</td>
<td>N1POL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>453</td>
</tr>
<tr>
<td>NCO1CLK</td>
<td>—</td>
<td>N1PWS&lt;2:0&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>N1CKS&lt;2:0&gt;</td>
<td></td>
<td>454</td>
<td></td>
</tr>
<tr>
<td>NCO1ACCL</td>
<td>—</td>
<td>—</td>
<td>NCO1ACC&lt;7:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>455</td>
<td></td>
</tr>
<tr>
<td>NCO1ACCH</td>
<td>—</td>
<td>—</td>
<td>NCO1ACC&lt;15:8&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>455</td>
<td></td>
</tr>
<tr>
<td>NCO1ACCU</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>NCO1ACC&lt;19:16&gt;</td>
<td>456</td>
<td></td>
</tr>
<tr>
<td>NCO1INCL</td>
<td>—</td>
<td>NCO1INC&lt;7:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>456</td>
<td></td>
</tr>
<tr>
<td>NCO1INCH</td>
<td>—</td>
<td>NCO1INC&lt;15:8&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>456</td>
<td></td>
</tr>
<tr>
<td>NCO1INCU</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>NCO1INC&lt;19:16&gt;</td>
<td>457</td>
</tr>
</tbody>
</table>

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 μA. 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**
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.

\[
\text{TOFFSET} = \frac{\text{asin} \left( \frac{\text{VCPINV}}{\text{VPEAK}} \right)}{2 \pi \cdot \text{Freq}}
\]

When External Voltage Source is relative to Vss:

\[
\text{TOFFSET} = \frac{\text{asin} \left( \frac{\text{VSS} - \text{VCPINV}}{\text{VPEAK}} \right)}{2 \pi \cdot \text{Freq}}
\]

When External Voltage Source is relative to Vdd:

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.

\[
\text{RPULLUP} = \frac{\text{RSERIES} \cdot (\text{VPULLUP} - \text{VCPINV})}{\text{VCPINV}}
\]

When External Signal is relative to Vss:

\[
\text{RPULLDOWN} = \frac{\text{RSERIES} \cdot \text{VCPINV}}{(\text{VDD} - \text{VCPINV})}
\]

When External Signal is relative to Vdd:
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, ΔT, is 4*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**

\[
R = R_{\text{SERIES}} \left( \frac{V_{\text{BIAS}}}{V_{\text{PEAK}} \sin \left( \pi F_{\text{Freq}} \left( \frac{\Delta T}{2} \right) \right)} - 1 \right)
\]

R is pull-up or pull-down resistor.

VBIAS is VPULLUP when R is pull-up or VDD when R is pull-down.

ΔT 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 ± 600 µA 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 ± 600 µA and the minimum is at least ± 100 µA, 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**

\[
R_{\text{SERIES}} = \frac{V_{\text{MAXPEAK}} + V_{\text{MINPEAK}}}{7 \times 10^{-4}}
\]

**29.8 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 ZCD Configuration bit is cleared, the ZCD circuit will be active at POR. When the 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 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 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 ZCD bit.
29.10 Register Definitions: ZCD Control

REGISTER 29-1: ZCDCON: ZERO-CROSS DETECT CONTROL REGISTER

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>U-0</th>
<th>R-x</th>
<th>R/W-0/0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEN</td>
<td>—</td>
<td>OUT</td>
<td>POL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>INTP</td>
</tr>
<tr>
<td>bit 7</td>
<td></td>
<td></td>
<td>bit 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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  **SEN**: Zero-Cross Detect Software Enable bit
This bit is ignored when ZCDSEN configuration bit is set.
1 = Zero-cross detect is enabled.
0 = Zero-cross detect is disabled. ZCD pin operates according to PPS and TRIS controls.

bit 6  **Unimplemented**: Read as ‘0’

bit 5  **OUT**: Zero-Cross Detect Data Output bit
ZCDPOL bit = 0:
1 = ZCD pin is sinking current
0 = ZCD pin is sourcing current
ZCDPOL bit = 1:
1 = ZCD pin is sourcing current
0 = ZCD pin is sinking current

bit 4  **POL**: Zero-Cross Detect Polarity bit
1 = ZCD logic output is inverted
0 = ZCD logic output is not inverted

bit 3-2  **Unimplemented**: Read as ‘0’

bit 1  **INTP**: Zero-Cross Detect Positive-Going Edge Interrupt Enable bit
1 = ZCDIF bit is set on low-to-high ZCD_output transition
0 = ZCDIF bit is unaffected by low-to-high ZCD_output transition

bit 0  **INTN**: Zero-Cross Detect Negative-Going Edge Interrupt Enable bit
1 = ZCDIF bit is set on high-to-low ZCD_output transition
0 = ZCDIF bit is unaffected by high-to-low ZCD_output transition

TABLE 29-1: SUMMARY OF REGISTERS ASSOCIATED WITH THE ZCD MODULE

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZCDCON</td>
<td>SEN</td>
<td>—</td>
<td>OUT</td>
<td>POL</td>
<td>—</td>
<td>—</td>
<td>INTP</td>
<td>INTN</td>
<td>462</td>
</tr>
</tbody>
</table>

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

See MD1CARH Register

See MD1SRC Register

See MD1CARL Register

Data Signal Modulator

CH<3:0>

CHPOL

MS<4:0>

SYNC

CHSYNC

D

MOD

CL<4:0>

CLPOL

CARL

CARH

PPS

S

RxYPPS

OPOL
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

Carrier Low (CARL)

Carrier High (CARH)

Modulator (BIT)

CHSYNC = 1
CLSYNC = 0

CHSYNC = 1
CLSYNC = 1

CHSYNC = 0
CLSYNC = 0

CHSYNC = 0
CLSYNC = 1

FIGURE 30-3: No Synchronization (CHSYNC = 0, CLSYNC = 0)

carrier_high

carrier_low

modulator

MDCHSYNC = 0
MDCLSYNC = 0

Active Carrier State

carrier_high carrier_low carrier_high carrier_low

FIGURE 30-4: Carrier High Synchronization (CHSYNC = 1, CLSYNC = 0)

carrier_high

carrier_low

modulator

MDCHSYNC = 1
MDCLSYNC = 0

Active Carrier State

carrier_high both carrier_low carrier_high both carrier_low
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.

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>Bit Name Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD1</td>
<td>MD1</td>
</tr>
</tbody>
</table>

**REGISTER 30-1: MD1CON0: MODULATION CONTROL REGISTER 0**

<table>
<thead>
<tr>
<th></th>
<th>R/W-0/0</th>
<th>U-0</th>
<th>R-0/0</th>
<th>R/W-0/0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN</td>
<td>—</td>
<td>OUT</td>
<td>OPOL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>BIT</td>
</tr>
</tbody>
</table>

**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. (1)

- **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 (2)
  - 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.

**Note 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**

<table>
<thead>
<tr>
<th>bit 7-6</th>
<th>bit 5</th>
<th>bit 4</th>
<th>bit 3-2</th>
<th>bit 1</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>U-0</td>
<td>U-0</td>
<td>R/W-0/0</td>
<td>CHPOL</td>
<td>CHSYNC</td>
<td>—</td>
</tr>
<tr>
<td>U-0</td>
<td>U-0</td>
<td>R/W-0/0</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>U-0</td>
<td>U-0</td>
<td>R/W-0/0</td>
<td>CLPOL</td>
<td>CLSYNC</td>
<td>bit 0</td>
</tr>
</tbody>
</table>

**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 (1)

**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 (1)

**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

| bit 7-5 | Unimplemented: Read as '0' |
| bit 4-0 | CH<4:0>: Modulator Carrier High Selection bits(1) |
| See Table 30-1 for signal list |
| Note 1: Unused selections provide an input value. |

REGISTER 30-4:  MD1CARL: MODULATION LOW CARRIER CONTROL REGISTER

| bit 7-5 | Unimplemented: Read as '0' |
| bit 4-0 | CL<4:0>: Modulator Carrier Low Input Selection bits(1) |
| 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

<table>
<thead>
<tr>
<th>CH&lt;4:0&gt; Connection</th>
<th>MD1CARH</th>
<th>MD1CARL Connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>11111-10011 31-19</td>
<td>Reserved</td>
<td>11111-10011 31-19</td>
</tr>
<tr>
<td>10010 18</td>
<td>CLC4OUT</td>
<td>10010 18 CLC4OUT</td>
</tr>
<tr>
<td>10001 17</td>
<td>CLC3OUT</td>
<td>10001 17 CLC3OUT</td>
</tr>
<tr>
<td>10000 16</td>
<td>CLC2OUT</td>
<td>10000 16 CLC2OUT</td>
</tr>
<tr>
<td>01111 15</td>
<td>CLC1OUT</td>
<td>01111 15 CLC1OUT</td>
</tr>
<tr>
<td>01110 14</td>
<td>NCO1OUT</td>
<td>01110 14 NCO1OUT</td>
</tr>
<tr>
<td>01101-01100 13-12</td>
<td>Reserved</td>
<td>01101-01100 13-12</td>
</tr>
<tr>
<td>01011 11</td>
<td>PWM8 OUT</td>
<td>01011 11 PWM8 OUT</td>
</tr>
<tr>
<td>01010 10</td>
<td>PWM7 OUT</td>
<td>01010 10 PWM7 OUT</td>
</tr>
<tr>
<td>01001 9</td>
<td>PWM6 OUT</td>
<td>01001 9 PWM6 OUT</td>
</tr>
<tr>
<td>01000 8</td>
<td>PWM5 OUT</td>
<td>01000 8 PWM5 OUT</td>
</tr>
<tr>
<td>00111 7</td>
<td>CCP4 OUT</td>
<td>00111 7 CCP4 OUT</td>
</tr>
<tr>
<td>00110 6</td>
<td>CCP3 OUT</td>
<td>00110 6 CCP3 OUT</td>
</tr>
<tr>
<td>00101 5</td>
<td>CCP2 OUT</td>
<td>00101 5 CCP2 OUT</td>
</tr>
<tr>
<td>00100 4</td>
<td>CCP1 OUT</td>
<td>00100 4 CCP1 OUT</td>
</tr>
<tr>
<td>00011 3</td>
<td>CLKREF output</td>
<td>00011 3 CLKREF output</td>
</tr>
<tr>
<td>00010 2</td>
<td>HFINTOSC</td>
<td>00010 2 HFINTOSC</td>
</tr>
<tr>
<td>00001 1</td>
<td>FOSC (system clock)</td>
<td>00001 1 FOSC (system clock)</td>
</tr>
<tr>
<td>00000 0</td>
<td>Pin selected by MD1CARHPPS</td>
<td>00000 0 Pin selected by MD1CARLPPS</td>
</tr>
</tbody>
</table>

### REGISTER 30-5: MD1SRC: MODULATION SOURCE CONTROL REGISTER

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>MS&lt;4:0&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 7</td>
<td>bit 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

- **Unimplemented**: Read as ‘0’
- **MS<4:0>**: Modulator Source Selection bits\(^{(1)}\)

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

<table>
<thead>
<tr>
<th>MS&lt;4:0&gt;</th>
<th>Connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1111</td>
<td>Reserved</td>
</tr>
<tr>
<td>-</td>
<td>31-23</td>
</tr>
<tr>
<td>1 0110</td>
<td>SPI1 SDO</td>
</tr>
<tr>
<td>1 0101</td>
<td>UART2 TX</td>
</tr>
<tr>
<td>1 0011</td>
<td>UART1 TX</td>
</tr>
<tr>
<td>1 0010</td>
<td>CLC4 OUT</td>
</tr>
<tr>
<td>1 0001</td>
<td>CLC3 OUT</td>
</tr>
<tr>
<td>1 0000</td>
<td>CLC2 OUT</td>
</tr>
<tr>
<td>0 1111</td>
<td>CLC1 OUT</td>
</tr>
<tr>
<td>0 1110</td>
<td>CMP2 OUT</td>
</tr>
<tr>
<td>0 1101</td>
<td>CMP1 OUT</td>
</tr>
<tr>
<td>0 1100</td>
<td>NCO1 OUT</td>
</tr>
<tr>
<td>0 1011</td>
<td>Reserved</td>
</tr>
<tr>
<td>0 1010</td>
<td>Reserved</td>
</tr>
<tr>
<td>0 1001</td>
<td>PWM8 OUT</td>
</tr>
<tr>
<td>0 1000</td>
<td>PWM7 OUT</td>
</tr>
<tr>
<td>0 0111</td>
<td>PWM6 OUT</td>
</tr>
<tr>
<td>0 0110</td>
<td>PWM5 OUT</td>
</tr>
</tbody>
</table>

### TABLE 30-3: SUMMARY OF REGISTERS ASSOCIATED WITH DATA SIGNAL MODULATOR MODE

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD1CON0</td>
<td>EN</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>469</td>
</tr>
<tr>
<td>MD1CON1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>470</td>
</tr>
<tr>
<td>MD1CARH</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>471</td>
</tr>
<tr>
<td>MD1CARL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>471</td>
</tr>
<tr>
<td>MDSRC</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>472</td>
</tr>
</tbody>
</table>

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**
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 LSBs 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.

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.

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.

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.

---

**Note:**

- The UxTXIF Transmitter Interrupt flag is set when the TXEN enable bit is set and the UxTXB register can accept data.

**Note:**

- The TSR is not mapped in data memory, so it is not available to the user.
### 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 RxHPPS 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**

- Write to UxTXB
- BRG Output (Shift Clock)
- TX pin
- UxTXIF bit (Transmit Buffer Reg. Empty Flag)
- TXMTIF bit (Transmit Shift Reg. Empty Flag)

---

**FIGURE 31-4: ASYNCHRONOUS TRANSMISSION (BACK-TO-BACK)**

- Write to UxTXB
- BRG Output (Shift Clock)
- TX pin
- UxTXIF bit (Transmit Buffer Reg. Empty Flag)
- TXMTIF bit (Transmit Shift Reg. Empty Flag)

**Note:** This timing diagram shows the first transmission and the start of the second consecutive transmission.
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:

- FERIE if FERIE is set
- PERIE 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.

© 2017 Microchip Technology Inc.

Preliminary DS40001919B-page 478
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 PIEx 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

Note: This timing diagram shows three words appearing on the RX input. The UxRXB (receive buffer) is not read before the third word is received, causing the RXFOIF (FIFO overrun) bit to be set. STPMD = 0, STP<1:0> = 00.
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 (UART1 only)

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 re-transmit 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:
- \( \text{MODE<3:0>} = 1010 \)
- \( \text{TXEN} = 1 \)
- \( \text{RXEN} = 0 \)
- \( \text{TXPOL} = 0 \)
- \( \text{UxP1} = \text{One less than the number of bytes to transmit (excluding the Start code)} \)
- \( \text{UxBRGH:L} = \text{Value to achieve 250K baud rate} \)
- \( \text{STP<1:0>} = 10 \) for 2 Stop bits
- \( \text{ON} = 1 \)
- \( \text{UxRXPPS} = \text{code for desired input pin} \)
- \( \text{Input pin ANSEL bit} = 0 \)

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 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 UxP2 and UxP3 registers. The UxP2 register holds the value of the byte number to start 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 UxP2H:L and 00Ch (12 decimal) to UxP3H:L. The receive FIFO is only 2 bytes deep, therefore the bytes must be retrieved by reading 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.

31.4.2 DMX RECEIVER

DMX Receiver mode is configured with the following settings:
- \( \text{MODE<3:0>} = 1010 \)
- \( \text{TXEN} = 0 \)
- \( \text{RXEN} = 1 \)
- \( \text{RXPOL} = 0 \)
- \( \text{UxP2} = \text{number of first byte to receive} \)
- \( \text{UxP3} = \text{number of last byte to receive} \)
- \( \text{UxBRGH:L} = \text{Value to achieve 250K baud rate} \)
- \( \text{STP<1:0>} = 10 \) for 2 Stop bits
- \( \text{ON} = 1 \)
- \( \text{UxRXPPS} = \text{code for desired input pin} \)
- \( \text{Input pin ANSEL bit} = 0 \)

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 UxP2 and UxP3 registers. The UxP2 register holds the value of the byte number to start 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 UxP2H:L and 00Ch (12 decimal) to UxP3H:L. The receive FIFO is only 2 bytes deep, therefore the bytes must be retrieved by reading UxRXB as they come in to avoid a receive FIFO overrun condition.

Since the DMX sequence always starts with a Break, the software can verify that it is in sync with the sequence by monitoring the RXBKIF flag to ensure that the next byte received after the RXBKIF is processed as the Start Code and subsequent bytes are processed as the expected data.
31.5 LIN Modes (UART1 only)

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 DACI Mode (UART1 only)

DACI 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 µs long. A double half-bit time or a single bit is typically 833.3 µs.

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 DACI 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. The receiver bit rate is determined by the BRG register. 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 the 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.

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** = 0b1000
- **TXEN** = 1
- **RXEN** = 1
- **UxP1** = Forward frames are held for transmission with 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** = 0b10 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. After sending the control byte, each data 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 that 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 an 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 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 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 (\( \approx \)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 expires, are 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 plus the UxP1 wait time.

The wait timer is reset by the backward frame and starts over immediately following the reception of 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. A new control byte can then be written to the UxTXB register. The control byte will be held in the buffer and sent at 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 to determine 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.

### CONTROL GEAR

The Control Gear mode is configured with the following settings:

- **MODE** = 0b1001
- **TXEN** = 1
- **RXEN** = 1
- **UxP1** = Back Frames are held for transmission this number of half-bit periods after the completion of a Forward Frame.
- **UxP2** = Forward/Back Frame threshold delimiter. Idle periods more than this number of half-bit periods are detected as Forward Frames.
- **UxBRGH:L** = Value to achieve 1200 baud rate
- **TXPOL** = appropriate polarity for interface circuit
- **RXPOL** = same as TXPOL
- **STP** = 0b10 for two Stop bits
- **RxyPPS** = TX pin output code
- **TX pin TRIS control** = 0
- **RXPPS** = RX pin selection code
- **RX pin TRIS control** = 1
- **Input pin ANSEL bit** = 0
- **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 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 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.
31.7 General Purpose Manchester  
(UART1 only)

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 = desired baud rate
- TXPOL and RXPOL = desired Idle state
- STP = desired number of stop periods
- RxyPPS = TX pin selection code
- TX pin TRIS control = 0
- RXPPS = 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 UXRXIF

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 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 RTS, and 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 RTS is an output and CTS is an input.

The RTS and CTS signals work as a pair to control the transmission flow. A DTE-to-DTE configuration connects the RTS output of the receiving UART to the CTS input of the sending UART. Refer to Figure 31-10.

The UART receiving data asserts the RTS output low when the input FIFO is empty. When a character is received, the RTS output goes high until the UxRXB is read to free up both FIFO locations.

When the 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 CTS input is high.

31.12.2 RS-485 TRANSEIVER 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.

Note 1: Configure UxCTSPPS to an unimplemented input such as RD0 (UxCTSPPS = 0x18).
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 (UART1 only)

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:

\[
\text{Desired Baud Rate} = \frac{F_{osc}}{16(UxBRG) + 1}
\]

\[
X = \frac{F_{osc}}{\text{Desired Baud Rate}} - 1
\]

\[
= \frac{16000000}{9600} - 1
\]

\[
= [103.17] = 103
\]

\[
\text{Calculated Baud Rate} = \frac{16000000}{16(103 + 1)}
\]

\[
= 9615
\]

\[
\text{Error} = \frac{\text{Calculated Baud Rate} - \text{Desired Baud Rate}}{\text{Desired Baud Rate}}
\]

\[
= \frac{9615 - 9600}{9600} = 0.16\%
\]

Table 31-1: Baud Rate Formulas

<table>
<thead>
<tr>
<th>BRGS</th>
<th>BRG/UART Mode</th>
<th>Baud Rate Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>High Rate</td>
<td>Fosc/[4 (n+1)]</td>
</tr>
<tr>
<td>0</td>
<td>Normal Rate</td>
<td>Fosc/[16(n+1)]</td>
</tr>
</tbody>
</table>

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.

**Note:** In DALI Mode, ABDEN is ignored. The baud rate needs to be manually set to 1200 using the BRG registers.

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 UxBRG 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>
<thead>
<tr>
<th>BRGS</th>
<th>BRG Base Clock</th>
<th>BRG ABD Clock</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Fosc/4</td>
<td>Fosc/32</td>
</tr>
<tr>
<td>0</td>
<td>Fosc/16</td>
<td>Fosc/128</td>
</tr>
</tbody>
</table>

**FIGURE 31-12: AUTOMATIC BAUD RATE CALIBRATION**

<table>
<thead>
<tr>
<th>BRG Value</th>
<th>RX pin</th>
<th>BRG Clock</th>
<th>ABDEN bit</th>
<th>RXIDL</th>
<th>ABDIF bit (Interrupt)</th>
<th>UxBRG</th>
</tr>
</thead>
<tbody>
<tr>
<td>XXXXh</td>
<td>Start</td>
<td>bit 0</td>
<td>bit 1</td>
<td>bit 2</td>
<td>bit 3</td>
<td>bit 4</td>
</tr>
<tr>
<td>0000h</td>
<td>bit 5</td>
<td>bit 6</td>
<td>bit 7</td>
<td>Stop</td>
<td></td>
<td></td>
</tr>
<tr>
<td>001Ch</td>
<td></td>
<td>Auto Cleared</td>
<td></td>
<td></td>
<td>Cleared by software</td>
<td></td>
</tr>
</tbody>
</table>

**Note 1:** Auto-baud is supported in LIN and 8-bit asynchronous modes only.
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 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 PIEx 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 UxE bit in the PIEx 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:

- UxE bit in the PIEx 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.
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 BRKOV 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.

SENDB 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 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

- Write to UxTXB
- BRG Output (Shift Clock)
- TX (pin)
- UxTXIF bit (Transmit Interrupt Flag)
- TXMTIF bit (Transmit Shift Empty Flag)
- SENDB (send Break control bit)

Sync start

Start bit bit 0 bit 1 bit 11 Stop bit

Break

Auto Cleared

Sync start
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.

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>Bit Name Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>UART 1</td>
<td>U1</td>
</tr>
<tr>
<td>UART 2</td>
<td>U2</td>
</tr>
</tbody>
</table>

REGISTER 31-1: UxCON0: UART CONTROL REGISTER 0

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/WHS/HC-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>MODE&lt;3:0&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRGS</td>
<td>ABDEN</td>
<td>TXEN</td>
<td>RXEN</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 \(\quad 0' = Bit is cleared\)
HC = Hardware clear

bit 7    BRGS: Baud rate Generator Speed Select bit
\(1 =\) Baud rate generator is high speed with 4 baud clocks per bit
\(0 =\) Baud rate generator is normal speed with 16 baud clocks per bit

bit 6    ABDEN: Auto-baud Detect Enable bit(3)
\(1 =\) Auto-baud is enabled. Receiver is waiting for Sync character (0x55)
\(0 =\) Auto-baud is not enabled or auto-baud is complete

bit 5    TXEN: Transmit Enable Control bit(2)
\(1 =\) Transmit is enabled. TX output pin drive is forced on when transmission is active, and controlled by PORT TRIS control when transmission is idle.
\(0 =\) Transmit is disabled. TX output pin drive is controlled by PORT TRIS control

bit 4    RXEN: Receive Enable Control bit(2)
\(1 =\) Receiver is enabled
\(0 =\) Receiver is disabled

bit 3-0   MODE<3:0>: UART Mode Select bits(1)

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111</td>
<td>Reserved</td>
</tr>
<tr>
<td>1110</td>
<td>Reserved</td>
</tr>
<tr>
<td>1101</td>
<td>Reserved</td>
</tr>
<tr>
<td>1100</td>
<td>LIN Master/Slave mode(4)</td>
</tr>
<tr>
<td>1011</td>
<td>LIN Slave-Only mode(4)</td>
</tr>
<tr>
<td>1010</td>
<td>DMX mode(4)</td>
</tr>
<tr>
<td>1001</td>
<td>DALI Control Gear mode(4)</td>
</tr>
<tr>
<td>1000</td>
<td>DALI Control Device mode(4)</td>
</tr>
<tr>
<td>0111</td>
<td>Reserved</td>
</tr>
<tr>
<td>0110</td>
<td>Reserved</td>
</tr>
<tr>
<td>0101</td>
<td>Reserved</td>
</tr>
<tr>
<td>0100</td>
<td>Asynchronous 9-bit UART Address mode. 9th bit: 1 = address, 0 = data</td>
</tr>
<tr>
<td>0011</td>
<td>Asynchronous 8-bit UART mode with 9th bit even parity</td>
</tr>
<tr>
<td>0010</td>
<td>Asynchronous 8-bit UART mode with 9th bit odd parity</td>
</tr>
<tr>
<td>0001</td>
<td>Asynchronous 7-bit UART mode</td>
</tr>
<tr>
<td>0000</td>
<td>Asynchronous 8-bit UART mode</td>
</tr>
</tbody>
</table>

Note 1: Changing the UART MODE while ON = 1 may cause unexpected results.
Note 2: Clearing TXEN or RXEN will not clear the corresponding buffers. Use TXBE or RXBE to clear the buffers.
Note 3: When MODE = 100x, then ABDEN bit is ignored.
Note 4: UART1 only.
REGISTER 31-2:  UxCON1: UART CONTROL REGISTER 1

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>R/W-0/0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W/HC-0/0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W/HC-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td>—</td>
<td>—</td>
<td>WUE</td>
<td>RXBIMD</td>
<td>—</td>
<td>BRKOVR</td>
<td>SENDB</td>
</tr>
</tbody>
</table>

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

- **ON**: Serial Port Enable bit
  - 1 = Serial port enabled
  - 0 = Serial port disabled (held in Reset)

- **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 PIEx to enable wake
  - 0 = Receiver operates normally

- **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

- **BRKOVR**: Send Break Software Override bit
  - 1 = TX output is forced to non-idle state
  - 0 = TX output is driven by transmit shift register

- **SENDB**: Send Break Control bit(1)
  - 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**

<table>
<thead>
<tr>
<th>bit</th>
<th>Description</th>
<th>Value at POR/BOR</th>
<th>Value at all other Resets</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>RUNOVF: Run During Overflow Control bit</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = RX input shifter continues to synchronize with Start bits after overflow condition</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = RX input shifter stops all activity on receiver overflow condition</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>RXPOL: Receive Polarity Control bit</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = Invert RX polarity, Idle state is low</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = RX polarity is not inverted, Idle state is high</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5-4</td>
<td>STP&lt;1:0&gt;: Stop Bit Mode Control bits(1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>11 = Transmit 2 Stop bits, receiver verifies first Stop bit</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10 = Transmit 2 Stop bits, receiver verifies first and second Stop bits</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>01 = Transmit 1.5 Stop bits, receiver verifies first Stop bit</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>00 = Transmit 1 Stop bit, receiver verifies first Stop bit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>C0EN: Checksum Mode Select bit(2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LIN mode:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = Checksum Mode 1, enhanced LIN checksum includes PID in sum</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = Checksum Mode 0, legacy LIN checksum does not include PID in sum</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Other modes:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = Add all TX and RX characters</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = Checksums disabled</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>TXPOL: Transmit Polarity Control bit</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = Output data is inverted, TX output is low in Idle state</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = Output data is not inverted, TX output is high in Idle state</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-0</td>
<td>FLO&lt;1:0&gt;: Handshake Flow Control bits</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>11 = Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10 = RTS/CTS and TXDE Hardware flow control</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>01 = XON/XOFF Software flow control</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>00 = Flow control is off</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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

**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.

**Note 2:** UART1 only.
## REGISTER 31-4:  UxERRIR: UART ERROR INTERRUPT FLAG REGISTER

<table>
<thead>
<tr>
<th></th>
<th>R/S/C-1/1</th>
<th>R/S/C-0/0</th>
<th>R/W/S-0/0</th>
<th>R/S/C-0/0</th>
<th>R/W/S-0/0</th>
<th>R/W/S-0/0</th>
<th>R/W/S-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TXMTIF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PERIF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ABDOVF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CERIF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FERIF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RXBKIF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RXFOIF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TXCIF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 7 TXMTIF: Transmit Shift Register Empty Interrupt Flag bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 = Transmit shift register is empty (Set at end of Stop bits)</td>
</tr>
<tr>
<td>0 = Transmit shift register is actively shifting data</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 6 PERIF: Parity Error Interrupt Flag bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIN and Parity modes:</td>
</tr>
<tr>
<td>1 = Unread byte at top of input FIFO has parity error</td>
</tr>
<tr>
<td>0 = Unread byte at top of input FIFO does not have parity error</td>
</tr>
<tr>
<td>DALI Device mode:</td>
</tr>
<tr>
<td>1 = Unread byte at top of input FIFO received as Forward Frame</td>
</tr>
<tr>
<td>0 = Unread byte at top of input FIFO received as Back Frame</td>
</tr>
<tr>
<td>Address mode:</td>
</tr>
<tr>
<td>1 = Unread byte at top of input FIFO received as address</td>
</tr>
<tr>
<td>0 = Unread byte at top of input FIFO received as data</td>
</tr>
<tr>
<td>Other modes:</td>
</tr>
<tr>
<td>Not used</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 5 ABDOVF: Auto-baud Detect Overflow Interrupt Flag bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>DALI mode:</td>
</tr>
<tr>
<td>1 = Start bit measurement overflowed counter</td>
</tr>
<tr>
<td>0 = No overflow during Start bit measurement</td>
</tr>
<tr>
<td>Other modes:</td>
</tr>
<tr>
<td>1 = Baud rate generator overflowed during the auto detection sequence</td>
</tr>
<tr>
<td>0 = Baud rate generator has not overflowed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 4 CERIF: Checksum Error Interrupt Flag bit (LIN mode only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 = Checksum error</td>
</tr>
<tr>
<td>0 = No checksum error</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 3 FERIF: Framing Error Interrupt Flag bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 = Unread byte at top of input FIFO has framing error</td>
</tr>
<tr>
<td>0 = Unread byte at top of input FIFO does not have framing error</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 2 RXBKIF: Break Reception Interrupt Flag bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 = Break detected</td>
</tr>
<tr>
<td>0 = No Break detected</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 1 RXFOIF: Receive FIFO Overflow Interrupt Flag bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 = Receive FIFO has overflowed</td>
</tr>
<tr>
<td>0 = Receive FIFO has not overflowed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 0 TXCIF: Transmit Collision Interrupt Flag bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 = Transmitted word is not equal to the word received during transmission</td>
</tr>
<tr>
<td>0 = Transmitted word equals the word received during transmission</td>
</tr>
</tbody>
</table>

---

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
### REGISTER 31-5: UxERRIE: UART ERROR INTERRUPT ENABLE REGISTER

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit 0</th>
<th>Bit 1</th>
<th>Bit 2</th>
<th>Bit 3</th>
<th>Bit 4</th>
<th>Bit 5</th>
<th>Bit 6</th>
<th>Bit 7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TXCIE</td>
<td>RXFOIE</td>
<td>RXBKIE</td>
<td>FERIE</td>
<td>CERIE</td>
<td>ABDVE</td>
<td>PERIE</td>
<td>TXMTIE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Legend:</th>
</tr>
</thead>
<tbody>
<tr>
<td>R = Readable bit</td>
</tr>
<tr>
<td>W = Writable bit</td>
</tr>
<tr>
<td>U = Unimplemented bit, read as ‘0’</td>
</tr>
<tr>
<td>u = Bit is unchanged</td>
</tr>
<tr>
<td>x = Bit is unknown</td>
</tr>
<tr>
<td>‘1’ = Bit is set</td>
</tr>
<tr>
<td>‘0’ = Bit is cleared</td>
</tr>
</tbody>
</table>

- **TXMTIE**: Transmit Shift Register Empty Interrupt Enable bit
  - 1 = Interrupt enabled
  - 0 = Interrupt not enabled
- **PERIE**: Parity Error Interrupt Enable bit
  - 1 = Interrupt enabled
  - 0 = Interrupt not enabled
- **ABDOVE**: Auto-baud Detect Overflow Interrupt Enable bit
  - 1 = Interrupt enabled
  - 0 = Interrupt not enabled
- **FERIE**: Framing Error Interrupt Enable bit
  - 1 = Interrupt enabled
  - 0 = Interrupt not enabled
- **RXBKIE**: Break Reception Interrupt Enable bit
  - 1 = Interrupt enabled
  - 0 = Interrupt not enabled
- **RXFOIE**: Receive FIFO Overflow Interrupt Enable bit
  - 1 = Interrupt enabled
  - 0 = Interrupt not enabled
- **TXCIE**: Transmit Collision Interrupt Enable bit
  - 1 = Interrupt enabled
  - 0 = Interrupt not enabled
## REGISTER 31-6: UxUIR: UART GENERAL INTERRUPT REGISTER

<table>
<thead>
<tr>
<th>Bit</th>
<th>WUIF</th>
<th>ABDIF</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>U-0</th>
<th>U-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5-3</td>
<td>Unimplemented</td>
<td>Unimplemented</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-0</td>
<td>Unimplemented</td>
<td>Unimplemented</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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: UxFIFO: UART FIFO STATUS REGISTER

<table>
<thead>
<tr>
<th>R/W/S-0/0</th>
<th>R/W-0/0</th>
<th>R/W/S/C-1/1</th>
<th>R/S/C-0/0</th>
<th>S/C-1/1</th>
<th>R/W/S/C-1/1</th>
<th>R/S/C-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TXWRE</td>
<td>STPMD</td>
<td>TXBE</td>
<td>TXBF</td>
<td>RXIDL</td>
<td>XON</td>
<td>RXBE</td>
</tr>
</tbody>
</table>

#### 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
- 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

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRG&lt;7:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRG&lt;15:8&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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: UxRX: UART RECEIVE REGISTER

<table>
<thead>
<tr>
<th>bit 7-0</th>
<th>RXB&lt;7:0&gt;:</th>
<th>Top of Receive Buffer</th>
</tr>
</thead>
</table>

Legend:
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as ‘0’
- u = Bit is unchanged
- x = Bit is unknown
- ’0’ = Bit is cleared
- -n/n = Value at POR and BOR/Value at all other Resets

REGISTER 31-11: UxTX: UART TRANSMIT REGISTER

<table>
<thead>
<tr>
<th>bit 7-0</th>
<th>TXB&lt;7:0&gt;:</th>
<th>Bottom of Transmit Buffer</th>
</tr>
</thead>
</table>

Legend:
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as ‘0’
- u = Bit is unchanged
- x = Bit is unknown
- ’0’ = Bit is cleared
- -n/n = Value at POR and BOR/Value at all other Resets
REGISTER 31-12: UxP1H: UART PARAMETER 1 HIGH REGISTER

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>P1&lt;8&gt;</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1&lt;7:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7-6</td>
<td>——</td>
<td>——</td>
<td>——</td>
<td>——</td>
<td>——</td>
<td>——</td>
<td>——</td>
<td>P2&lt;8&gt;</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>P2&lt;7:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>bit 7-0</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>P3&lt;8&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>——</td>
<td>——</td>
<td>——</td>
<td>——</td>
<td>——</td>
<td>——</td>
<td>——</td>
<td>——</td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>bit 7-0</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>P3&lt;7:0&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TXCH&lt;7:0&gt;</td>
<td>Checksum calculated from TX bytes</td>
</tr>
<tr>
<td><strong>Legend:</strong></td>
<td></td>
</tr>
<tr>
<td>R = Readable bit</td>
<td>W = Writable bit</td>
</tr>
<tr>
<td>u = Bit is unchanged</td>
<td>x = Bit is unknown</td>
</tr>
<tr>
<td>‘1’ = Bit is set</td>
<td>‘0’ = Bit is cleared</td>
</tr>
</tbody>
</table>

#### Bit 7-0
- **TXCH<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

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RXCH&lt;7:0&gt;</td>
<td>Checksum calculated from RX bytes</td>
</tr>
<tr>
<td><strong>Legend:</strong></td>
<td></td>
</tr>
<tr>
<td>R = Readable bit</td>
<td>W = Writable bit</td>
</tr>
<tr>
<td>u = Bit is unchanged</td>
<td>x = Bit is unknown</td>
</tr>
<tr>
<td>‘1’ = Bit is set</td>
<td>‘0’ = Bit is cleared</td>
</tr>
</tbody>
</table>

#### Bit 7-0
- **RXCH<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

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on page</th>
</tr>
</thead>
<tbody>
<tr>
<td>UxCON0</td>
<td>BRGS</td>
<td>ABDEN</td>
<td>TXEN</td>
<td>RXEN</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>MODE&lt;3:0&gt;</td>
</tr>
<tr>
<td>UxCON1</td>
<td>ON</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>WUE</td>
<td>RXBIMD</td>
<td>—</td>
<td>—</td>
<td>BRKOVR</td>
</tr>
<tr>
<td>UxCON2</td>
<td>RUNOFV</td>
<td>RXPOL</td>
<td>STP&lt;1:0&gt;</td>
<td>COEN</td>
<td>TXPOL</td>
<td>FLO&lt;1:0&gt;</td>
<td>—</td>
<td>—</td>
<td>SENDB</td>
</tr>
<tr>
<td>UxERRIR</td>
<td>TXMTIF</td>
<td>PERIF</td>
<td>ABDOVF</td>
<td>CERIF</td>
<td>FERIF</td>
<td>RXBKIF</td>
<td>RXFOIF</td>
<td>TCCIF</td>
<td>501</td>
</tr>
<tr>
<td>UxERRIE</td>
<td>TXMTIE</td>
<td>PERIE</td>
<td>ABDOVE</td>
<td>CERIE</td>
<td>FERIE;</td>
<td>RXBKIE</td>
<td>RXFOIE</td>
<td>TXCIE</td>
<td>502</td>
</tr>
<tr>
<td>UxUIR</td>
<td>WUIF</td>
<td>ABDIF</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ABDE</td>
<td>—</td>
<td>—</td>
<td>503</td>
</tr>
<tr>
<td>UxFIFO</td>
<td>TXWRE</td>
<td>STPMD</td>
<td>TXBE</td>
<td>TXBF</td>
<td>RXIDL</td>
<td>XON</td>
<td>RXBE</td>
<td>RXBF</td>
<td>504</td>
</tr>
<tr>
<td>UxBRGL</td>
<td>BRG&lt;7:0&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>505</td>
</tr>
<tr>
<td>UxBRGH</td>
<td>BRG&lt;15:8&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>506</td>
</tr>
<tr>
<td>UxRXB</td>
<td>RXB&lt;7:0&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>507</td>
</tr>
<tr>
<td>UxTXB</td>
<td>TXB&lt;7:0&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>508</td>
</tr>
<tr>
<td>UxP1H</td>
<td>P1&lt;7:0&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>509</td>
</tr>
<tr>
<td>UxP2H</td>
<td>P2&lt;7:0&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>510</td>
</tr>
<tr>
<td>UxP3H</td>
<td>P3&lt;7:0&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>510</td>
</tr>
<tr>
<td>UxP3L</td>
<td>P3&lt;7:0&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>510</td>
</tr>
<tr>
<td>UxTXCHK</td>
<td>TXCHK&lt;7:0&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>510</td>
</tr>
</tbody>
</table>

**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 (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.
FIGURE 32-1: SPI MODULE SIMPLIFIED BLOCK DIAGRAM

Note 1: If TXR=1 and the transmit FIFO is empty, the previous value of the receive shift register will be sent to the transmit serializer.
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)F2X/4X/5XK42 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 PIC18F2X/4X/5XK42 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

**Note 1:** In some modes, if the Transmit FIFO is empty, the most recently received byte of data will be transmitted.

**2:** This diagram assumes that the LSBF bit is cleared (communications are MSb-first). If LSBF is set, the communications will be LSb-first.
### 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 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 SPIxRxR 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 TXBF 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 the slave 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, SSS(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.

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.
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.

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 ≠ 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 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: 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.

32.4.2 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 ≠ 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 SPIx-TWIDTH ≠ 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>
<thead>
<tr>
<th>RXR = 1</th>
<th>TXR = 1</th>
<th>TXR = 0</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Full Duplex Mode</strong></td>
<td>If BMODE = 1, transfer when RxFIFO is not full and TxFIFO is not empty. If BMODE = 0, Transfer when RXFIFO is not full, TXFIFO is not empty, and the Transfer Counter is non-zero.</td>
<td>If BMODE = 0, Transfer when RxFIFO is not full and the Transfer Counter is non-zero. Transmitted data is either the top of the FIFO or the most recently received data.</td>
</tr>
<tr>
<td><strong>Receive Only mode</strong></td>
<td>If BMODE = 1, transfer when RxFIFO is not full and the Transfer Counter is non-zero.</td>
<td>No Transfers</td>
</tr>
<tr>
<td><strong>Transmit Only Mode</strong></td>
<td>If BMODE = 1, transfer when TxFIFO is not empty. If BMODE = 0, Transfer when TXFIFO is not empty and the Transfer Counter is non-zero. Received data is not stored.</td>
<td></td>
</tr>
<tr>
<td><strong>No Transfers</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TABLE 32-1: MASTER MODE TXR/RXR SETTINGs
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 SPIxTxXB 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

Note: 1. SS(out) is not shown on this diagram
2. SPIxTCNT write is optional TXR/RXR = 1/1 and BMODE=1. If BMODE=0, a write to SPIxTCNT is required to start transmission; TCZIF signals the end of the transmission.
3. Transmission gap occurs while waiting for transmitter data.
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

Note 1: SS(out) is not shown
2. The byte counter is optional when TXR/RXR = 1/0;
3. After the command bytes, wait for SRMTIF before loading SPIxBYTESL otherwise the command bytes would decrement BYTES. Alternatively, load BC = 5 and count the command bytes also; TCZIF signals the end of the transmission.
4. Transmit data interrupt handler (or DMA) must write only the bytes necessary; the byte counter is not available as an indicator.
5. Reading the RXFIFO is not required because RXR = 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**

\[
F_{BAUD} = \frac{F_{CSEL}}{2 \cdot (BAUD + 1)}
\]

where \( F_{BAUD} \) is the baud rate frequency output on the SCK pin, \( F_{CSEL} \) 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
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**

<table>
<thead>
<tr>
<th>TRISxn(1)</th>
<th>TXR</th>
<th>SS</th>
<th>TXBE</th>
<th>SDO State</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>FALSE</td>
<td>0</td>
<td>Drives state determined by LATxn(2)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>FALSE</td>
<td>1</td>
<td>Drives state determined by LATxn(2)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>TRUE</td>
<td>0</td>
<td>Outputs the oldest byte in the TXFIFO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Does not remove data from the TXFIFO</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>TRUE</td>
<td>1</td>
<td>Outputs the most recently received byte</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>FALSE</td>
<td>0</td>
<td>Drives state determined by LATxn(2)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>FALSE</td>
<td>1</td>
<td>Drives state determined by LATxn(2)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>TRUE</td>
<td>0</td>
<td>Outputs the oldest byte in the TXFIFO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Removes transmitted byte from the TXFIFO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Decrements occupancy of TXFIFO</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>TRUE</td>
<td>1</td>
<td>Outputs the most recently received byte</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Sets the TXUIF bit of SPIxINTF</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>FALSE</td>
<td>0</td>
<td>Tri-stated</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>FALSE</td>
<td>1</td>
<td>Tri-stated</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>TRUE</td>
<td>0</td>
<td>Tri-stated</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>TRUE</td>
<td>1</td>
<td>Tri-stated</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>FALSE</td>
<td>0</td>
<td>Tri-stated</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>FALSE</td>
<td>1</td>
<td>Tri-stated</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>TRUE</td>
<td>0</td>
<td>Outputs the oldest byte in the TXFIFO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Removes transmitted byte from the TXFIFO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Decrements occupancy of TXFIFO</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>TRUE</td>
<td>1</td>
<td>Outputs the most recently received byte</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Sets the TXUIF bit of SPIxINTF</td>
</tr>
</tbody>
</table>

**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 Section 32.8 “SPI Interrupts” the transfer counter in slave mode as well as Section X.8 for more information on interrupts).

FIGURE 32-11: SPI SLAVE MODE OPERATION – INTERRUPT-DRIVEN, MASTER WRITES 2+3 BYTES

Note: 1. This delay is exaggerated for illustration, and can be as short as 1/2 bit period.
2. If the device is sleeping, SOSIF will wake it up for interrupt service.
3. Setting SPIxTCNTL is optional in this example, otherwise it will count -3, -4, -5, and TCZIF will not occur.
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 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 PIR1 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 PIR1 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, SPIxTCT H/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.

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.

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.

FIGURE 32-14: TRANSFER AND SLAVE SELECT INTERRUPT TIMINGS

---

Note 1: SRMTIF available only in Master mode
Note 2: Clearing of interrupt flags is shown for illustration; actual interrupt flags must be cleared in software
Note 3: SOSIF and EOSIF are set according to SS(in), even in Master mode.
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

<table>
<thead>
<tr>
<th>R/W/HS-0/0</th>
<th>R/W/HS-0/0</th>
<th>R/W/HS-0/0</th>
<th>R/W/HS-0/0</th>
<th>U-0</th>
<th>R/W/HS-0/0</th>
<th>R/W/HS-0/0</th>
<th>U-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRMTIF</td>
<td>TCZIF</td>
<td>SOSIF</td>
<td>EOSIF</td>
<td></td>
<td>RXOIF</td>
<td>TXUIF</td>
<td></td>
</tr>
</tbody>
</table>

Legend:

- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as '0'
- **HS** = Bit can be set by hardware

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Slave mode:</th>
<th>Master mode:</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td><strong>SRMTIF</strong>: Shift Register Empty Interrupt Flag bit</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Slave mode:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>This bit is ignored</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Master mode:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = The data transfer is complete</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = Either no data transfers have occurred or a data transfer is in progress</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td><strong>TCZIF</strong>: Transfer Counter is Zero Interrupt Flag bit</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = The transfer counter (as defined by BMODE in Register 32-7, TCNTH/L, and TWIDTH) has decremented to zero</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = No interrupt pending</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td><strong>SOSIF</strong>: Start of Slave Select Interrupt Flag bit</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = SS(in) transitioned from false to true</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = No interrupt pending</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td><strong>EOSIF</strong>: End of Slave Select Interrupt Flag bit</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = SS(in) transitioned from true to false</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = No interrupt pending</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td><strong>Unimplemented</strong>: Read as '0'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td><strong>RXOIF</strong>: Receiver Overflow Interrupt Flag bit</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = Data transfer completed when RXBF = 1 (edge triggered) and RXR = 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = No interrupt pending</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td><strong>TXUIF</strong>: Transmitter Underflow Interrupt Flag bit</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = Slave Data transfer started when TXBE = 1 and TXR = 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = No interrupt pending</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td><strong>Unimplemented</strong>: Read as '0'</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**REGISTER 32-2: SPIxINTE: SPI INTERRUPT ENABLE REGISTER**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Function</th>
<th>R/W</th>
<th>Value</th>
<th>SRMTIE</th>
<th>TCZIE</th>
<th>SOSIE</th>
<th>EOSIE</th>
<th>RXOIE</th>
<th>TXUIE</th>
<th>U-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Shift Register Empty Interrupt Enable</td>
<td>R/W</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Transfer Counter is Zero Interrupt</td>
<td>R/W</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Start of Slave Select Interrupt</td>
<td>R/W</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>End of Slave Select Interrupt</td>
<td>R/W</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Unimplemented</td>
<td>R/W</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Receiver Overflow Interrupt</td>
<td>R/W</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Transmitter Underflow Interrupt</td>
<td>R/W</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>Unimplemented</td>
<td>R/W</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as ‘0’

**REGISTER 32-3: SPIxTCNTL – SPI TRANSFER COUNTER LSB REGISTER**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Function</th>
<th>R/W</th>
<th>Value</th>
<th>TCNT7</th>
<th>TCNT6</th>
<th>TCNT5</th>
<th>TCNT4</th>
<th>TCNT3</th>
<th>TCNT2</th>
<th>TCNT1</th>
<th>TCNT0</th>
</tr>
</thead>
<tbody>
<tr>
<td>7-0</td>
<td>TCNT&lt;7:0&gt;</td>
<td>R/W</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>U-0</td>
<td>U-0</td>
</tr>
<tr>
<td>U-0</td>
<td>U-0</td>
</tr>
<tr>
<td>U-0</td>
<td>U-0</td>
</tr>
<tr>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
</tr>
</tbody>
</table>

| TCNT10 | TCNT9 | TCNT8 |

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

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>U-0</td>
<td>U-0</td>
</tr>
<tr>
<td>U-0</td>
<td>U-0</td>
</tr>
<tr>
<td>U-0</td>
<td>U-0</td>
</tr>
<tr>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
</tr>
</tbody>
</table>

| TWIDTH2 | TWIDTH1 | TWIDTH0 |

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

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAUD7</td>
<td>BAUD6</td>
<td>BAUD5</td>
<td>BAUD4</td>
<td>BAUD3</td>
<td>BAUD2</td>
<td>BAUD1</td>
<td>BAUD0</td>
</tr>
</tbody>
</table>

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 Clock Period*(BAUD+1)
- SCK toggle frequency: FSCK=FBAUD= SPI 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

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN</td>
<td>LSBF</td>
<td>MST</td>
<td>BMODE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>U-0</th>
<th>R/W-1/1</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMP</td>
<td>CKE</td>
<td>CKP</td>
<td>FST</td>
<td>—</td>
<td>SSP</td>
<td>SDIP</td>
</tr>
<tr>
<td>bit 7</td>
<td>bit 0</td>
<td>bit 1</td>
<td>bit 2</td>
<td>bit 3</td>
<td>bit 4</td>
<td>bit 5</td>
</tr>
</tbody>
</table>

**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**

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 6</th>
<th>bit 5-3</th>
<th>bit 2</th>
<th>bit 1</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUSY</td>
<td>SSFLT</td>
<td></td>
<td>SSET</td>
<td>TXR</td>
<td>RXR</td>
</tr>
</tbody>
</table>

**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\(^{(1)}\)
  - 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\(^{(1)}\)
  - 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.

**Note 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

<table>
<thead>
<tr>
<th>R/C/HS-0/0</th>
<th>U-0</th>
<th>R-1/1</th>
<th>U-0</th>
<th>R/C/HS-0/0</th>
<th>S-0/0</th>
<th>U-0</th>
<th>R-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TXWE</td>
<td>—</td>
<td>TXBE</td>
<td>—</td>
<td>RXRE</td>
<td>CLRB</td>
<td>—</td>
<td>RXBF</td>
</tr>
</tbody>
</table>

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
**CLRB**: 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

<table>
<thead>
<tr>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>RXB7</td>
<td>RXB6</td>
<td>RXB5</td>
<td>RXB4</td>
<td>RXB3</td>
<td>RXB2</td>
<td>RXB1</td>
<td>RXB0</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>W-0</th>
<th>W-0</th>
<th>W-0</th>
<th>W-0</th>
<th>W-0</th>
<th>W-0</th>
<th>W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TXB7</td>
<td>TXB6</td>
<td>TXB5</td>
<td>TXB4</td>
<td>TXB3</td>
<td>TXB2</td>
<td>TXB1</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CLKSEL3</td>
<td>CLKSEL2</td>
<td>CLKSEL1</td>
<td>CLKSEL0</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPIxINTF</td>
<td>SRMTIF</td>
<td>TCZIF</td>
<td>SOSIF</td>
<td>EOSIF</td>
<td>—</td>
<td>RXOIF</td>
<td>TXUIF</td>
<td>—</td>
<td>535</td>
</tr>
<tr>
<td>SPIxINTE</td>
<td>SRMTIE</td>
<td>TCZIE</td>
<td>SOSIE</td>
<td>EOSIE</td>
<td>—</td>
<td>RXOIE</td>
<td>TXUIE</td>
<td>—</td>
<td>536</td>
</tr>
<tr>
<td>SPIxTCNTH</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>TCNT10</td>
<td>TCNT9</td>
<td>TCNT8</td>
<td>537</td>
</tr>
<tr>
<td>SPIxTCNTL</td>
<td>TCNT7</td>
<td>TCNT6</td>
<td>TCNT5</td>
<td>TCNT4</td>
<td>TCNT3</td>
<td>TCNT2</td>
<td>TCNT1</td>
<td>TCNT0</td>
<td>536</td>
</tr>
<tr>
<td>SPIxTWIDTH</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>TWIDTH2</td>
<td>TWIDTH1</td>
<td>TWIDTH0</td>
</tr>
<tr>
<td>SPIxBAUD</td>
<td>BAUD7</td>
<td>BAUD6</td>
<td>BAUD5</td>
<td>BAUD4</td>
<td>BAUD3</td>
<td>BAUD2</td>
<td>BAUD1</td>
<td>BAUD0</td>
<td>538</td>
</tr>
<tr>
<td>SPIxCON0</td>
<td>EN</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>LSBF</td>
<td>MST</td>
<td>BMODE</td>
</tr>
<tr>
<td>SPIxCON1</td>
<td>SMP</td>
<td>CKE</td>
<td>CKP</td>
<td>FST</td>
<td>—</td>
<td>—</td>
<td>SSP</td>
<td>SDIP</td>
<td>SDOP</td>
</tr>
<tr>
<td>SPIxCON2</td>
<td>BUSY</td>
<td>SSFLT</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>SSET</td>
<td>TXR</td>
<td>RXR</td>
</tr>
<tr>
<td>SPIxSTATUS</td>
<td>TXWE</td>
<td>—</td>
<td>TXBE</td>
<td>—</td>
<td>RXRE</td>
<td>CLRBF</td>
<td>—</td>
<td>RXBF</td>
<td>541</td>
</tr>
<tr>
<td>SPIxRXB</td>
<td>RXB7</td>
<td>RXB6</td>
<td>RXB5</td>
<td>RXB4</td>
<td>RXB3</td>
<td>RXB2</td>
<td>RXB1</td>
<td>RXB0</td>
<td>541</td>
</tr>
<tr>
<td>SPIxFXB</td>
<td>TXB7</td>
<td>TXB6</td>
<td>TXB5</td>
<td>TXB4</td>
<td>TXB3</td>
<td>TXB2</td>
<td>TXB1</td>
<td>TXB0</td>
<td>542</td>
</tr>
<tr>
<td>SPIxCLK</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CLKSEL3</td>
<td>CLKSEL2</td>
<td>CLKSEL1</td>
<td>CLKSEL0</td>
</tr>
</tbody>
</table>

**Legend:** — = unimplemented, read as ‘0’. Shaded cells are unused by the SPI module.
33.0 I²C MODULE

The device has two dedicated, independent I²C modules. Figure 33-1 is a block diagram of the I²C interface module. The figure shows both the Master and Slave modes together.

FIGURE 33-1: I²C MODULE BLOCK DIAGRAM
33.1 I^2C Features

- Inter-Integrated Circuit (I\(^2\)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\(^2\)C, SMBus 2.0, and SMBus 3.0 input level selections

33.2 I\(^2\)C Module Overview

The I\(^2\)C module provides a synchronous interface between the microcontroller and other I\(^2\)C-compatible devices using the two-wire I\(^2\)C serial bus. Devices communicate in a master/slave environment. The I\(^2\)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\(^2\)C bus has to be initiated by the Master.

Figure 33-2 shows a typical connection between a master and more than one slave.

![I2C MASTER/SLAVE CONNECTIONS](image-url)
There are four main operations based on the direction of the data being shared during I²C 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 I²C 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 I²C module are discussed in the section below.

### 33.3 I²C Mode Operation

All I²C 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 I²C devices.

#### 33.3.1 DEFINITION OF I²C TERMINOLOGY

The I²C 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 I²C specification.

<table>
<thead>
<tr>
<th>TERM</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmitter</td>
<td>The device which shifts data out onto the bus</td>
</tr>
<tr>
<td>Receiver</td>
<td>The device which shifts data in from the bus</td>
</tr>
<tr>
<td>Master</td>
<td>The device that initiates a transfer, generates clock signals and terminates a transfer</td>
</tr>
<tr>
<td>Slave</td>
<td>The device addressed by the master</td>
</tr>
<tr>
<td>Multi-master</td>
<td>A bus with more than one device that can initiate data transfers</td>
</tr>
<tr>
<td>Arbitration</td>
<td>Procedure to ensure that only one master at a time controls the bus. Winning arbitration ensures that the message is not corrupted</td>
</tr>
<tr>
<td>Synchronization</td>
<td>Procedure to synchronize the clocks of two or more devices on the bus.</td>
</tr>
<tr>
<td>Idle</td>
<td>No master is controlling the bus, and both SDA and SCL lines are high.</td>
</tr>
<tr>
<td>Active</td>
<td>Any time one or more master devices are controlling the bus.</td>
</tr>
<tr>
<td>Addressed Slave</td>
<td>Slave device that has received a matching address and is actively being clocked by a master</td>
</tr>
<tr>
<td>Matching Address</td>
<td>Address byte that is clocked into a slave that matches the value stored in I2CxADR</td>
</tr>
<tr>
<td>Write Request</td>
<td>Slave receives a matching address with R/W bit clear and is ready to clock in data</td>
</tr>
<tr>
<td>Read Request</td>
<td>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.</td>
</tr>
<tr>
<td>Clock Stretching</td>
<td>When a device on the bus holds SCL low to stall communication</td>
</tr>
<tr>
<td>Bus Collision</td>
<td>Any time the SDA line is sampled low by the module while it is outputting and expected high state.</td>
</tr>
<tr>
<td>Bus Timeout</td>
<td>Any time the I2CBTOISM input transitions high, the I²C module is reset and the module goes idle.</td>
</tr>
</tbody>
</table>
33.3.2 BYTE FORMAT

All communication in I²C 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 inputs. This is done by clearing the appropriate TRIS bits and setting the appropriate and ODCON bits. The user may also select the input threshold, slew-rate and internal pull-up settings using the RxI2C 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 I²C 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**

![Waveforms for Start and Stop Conditions](image)

**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.

33.3.8 ACKNOWLEDGE SEQUENCE
The ninth SCL pulse for any transferred byte in \( \text{I}^2\text{C} \) 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 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 ACK data being sent to the transmitter depends on the value of I2CxCNT register. ACKDT is the value sent when I2CxCNT! = 0. When I2CxCNT = 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 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 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 timeout source for the module. The \( \text{I}^2\text{C} \) 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 timeout 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.

Note: The bus time-out source should produce a rising edge.

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 I²C 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 I²C MODE**

<table>
<thead>
<tr>
<th>Modes</th>
<th>MODE&lt;2:0&gt;</th>
<th>I2CxADB0</th>
<th>I2CxADB1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slave (7-bit)</td>
<td>000</td>
<td>RX</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>001</td>
<td>RX</td>
<td>—</td>
</tr>
<tr>
<td>Slave (10-bit)</td>
<td>010</td>
<td>RX</td>
<td>RX</td>
</tr>
<tr>
<td></td>
<td>011</td>
<td>RX</td>
<td>RX</td>
</tr>
<tr>
<td>Master (7-bit)</td>
<td>100</td>
<td>—</td>
<td>TX</td>
</tr>
<tr>
<td>Master (10-bit)</td>
<td>101</td>
<td>TX</td>
<td>TX</td>
</tr>
<tr>
<td>Multi-Master (7-bit)</td>
<td>110</td>
<td>RX</td>
<td>TX</td>
</tr>
<tr>
<td></td>
<td>111</td>
<td>RX</td>
<td>TX</td>
</tr>
</tbody>
</table>

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 I²C 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 CSTRDIS (clock stretching disable) bit in the I2CxCON2 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 TXUF 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 I2CxCNT register is used to specify the number of bytes in a complete \( \text{i}^2\text{C} \) packet. The value in this register will decrement every time a data byte is received or transmitted from the \( \text{i}^2\text{C} \) module. The I2CxCNT register will not decrement past zero.

If a byte transfer causes the I2CxCNT register to decrement to zero, the Count Interrupt Flag bit (CNTIF) in I2CxPIR is set. This flag bit is set on the 9th falling edge of SCL for transmit operations and can be more informational if the edge is specified.

The I2CxCNT 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 I2CxCNT register.

Note 1: I2CxCNT 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 I2CxCNT register can be updated mid-message to prevent decrement to zero.

33.4 \( \text{i}^2\text{C} \) Slave Mode

The \( \text{i}^2\text{C} \) 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 I2CADB0/1 registers. If the value does not match, there is no response from the module. The \( \text{i}^2\text{C} \) 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.

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 I2C 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 ACK. The general call address is a reserved address in the I2C 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/W bit of the I2CxSTAT0 register.

33.4.3.1 Slave Reception (7-bit Addressing Mode)

This section describes the sequence of events for the I2C module configured as an I2C 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/A bit is cleared. If the address does not match; module becomes idle.
6. The matched address data is loaded into I2CxADB0 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 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/A 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 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: \textsuperscript{2}C SLAVE, 7-BIT ADDRESS, RECEPTION (ACKTIE = 0, ADRIE = 0, WRIE = 0)
FIGURE 33-7: \( \text{i}^2\text{C SLAVE, 7-BIT ADDRESS, RECEPTION WITH i2CxCNT (ACKTIE = 1, ADRIE = 0, WRIE = 0)} \)

- **From Slave to Master**: Bus Master sends stop condition.
- **Receive Address**:
  - SDA: A7 A6 A5 A4 A3 A2 A1 R/W = 0 ACK
  - SCL: A A A A A A A A
- **Receive Data**:
  - SDA: D7 D6 D5 D4 D3 D2 D1 D0
  - SCL: D A A A A A A A
- **I2CxCNT**:
  - Value for I2CxCNT = 0.
- **CSTR**:
  - SCIF is set
  - ADIHE = 0
  - ACKTIE = 1
  - Software clears CSTR
  - WRIE = 0
  - ACKTIE = 1
  - No CSTR for NACK
  - WRIE = 0
  - WRIF is set
  - ACKTIF is set
  - WRIF is set
  - ACKTIF is set
  - WRIF is set
  - ACKTIF is set
- **R/W**:
  - R/W copied from matching address
  - ADRIF is set
  - ACKTIF is set
  - WRIF is set
  - ACKTIF is set
- **D/A**:
  - matching address copied to I2CxADB0
- **I2CxCNT**:
  - 0x02
  - 0x02
  - 0x01
  - 0x00
  - Slave sends ACKCNT
  - value for I2CxCNT = 0
  - CNTIF is set
- **RXBF**:
  - Data byte written to I2CxRXB
  - RXIF set
  - Software reads I2CxRXB
  - clear RXIF
  - Second data byte written to I2CxRXB
  - RXIF set
FIGURE 33-8: I2C SLAVE, 7-BIT ADDRESS, RECEPTION NO I2CxCNT (ACKTIE = 0, ADRIE = 1, WRIE = 1)

Master Releases SDA to slave for ACK sequence

SDA

S

A7 A6 A5 A4 A3 A2 A1

ACK D7 D6 D5 D4 D3 D2 D1 D0

SDA

S

A7 A6 A5 A4 A3 A2 A1

ACK D7 D6 D5 D4 D3 D2 D1 D0

Receiving Address R/W = 0

Receiving Data

ACKT cleared by hardware in 9th rising edge of SCL

ACKT cleared by hardware in 8th falling edge of SCL

CSTR cleared software, SCL is released

Software clears CSTR SCL is released

ACK

Master send stop condition

R/W

R/W copied from matching address

D/A

matching address written to I2CxADB0

ACKDT

Software reads I2CxADB0

Software clears ACKDT

ACK

ACKT set by hardware on 8th falling edge of SCL

I2CxCNT

0x44

0x44

0x43

0x42

RXBF

I2CxRXIF is set

I2CxRXIF is set

I2CxRXBF is set

I2CxRXBF is set

Clearing I2CxRXIF

Clearing I2CxRXIF

Master Releases SDA to slave for ACK sequence

Clock is held low until CSTR is set to '1'

S

CSTR

CSTR cleared software, SCL is released

WRIE = 1

WRIE = 1

WRIF is set

WRIF is set

ACKTIF is set

NACKIF is set

Slave software sets ACKDT

Software reads data from I2CxRB

Clearing I2CxRXIF

Software clears ACKDT
33.4.3.2 Slave Transmission (7-bit Addressing Mode)

This section describes the sequence of events for the I²C module configured as an I²C 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/W bit, D/A bit is cleared. If the address does not match; module becomes idle.
6. The matched address data is loaded into I2CxADB0 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.
FIGURE 33-9: \( \text{I}^2 \text{C SLAVE, 7-BIT ADDRESS, TRANSMISSION} \)

- **Master Releases SDA**
- **Slave sends ACK**
- **Slave Transmitting Data**
- **Master sends ACK**
- **Master sends NACK**
- **Master sends stop condition**

**SDA**
- \( A7, A6, A5, A4, A3, A2, A1 \)
- \( D7, D6, D5, D4, D3, D2, D1, D0 \)

**SCL**
- \( P \)
- \( S \)

**ACKSTAT**
- **ACKTIE = 0**
- **ACKTIF is set**
- **NACKIF is set**

**I2CxCNT**
- **Software writes I2CxCNT**
- **0x02**
- **0x01**
- **0x00**
- **CNTIF is set**

**TXBE**
- **TXIF set for read address when TXBE = 1**
- **value in I2CxCNT is ignored**
- **Software writes I2CxB TXB**
- **MSb is of I2CxB TXB copied to SDA**
- **Data byte 1 loaded from I2CxB TXB to shifter**
- **TXIF set**
- **TXIF NOT set**

**ADRIE = 1**
- **SCIF is set**
- **Software write to I2CxB TXB clears CSTR**

**ADRIE = 0**
- **ACKTIE = 0**
- **ACKTIF is set**
- **PCIF is set**

**D/A**
- **R/W copied from matching address**
- **ADRF is set**

**R/W**
- **R/W copied from matching address**
- **ADRF is set**

**CSTR**
- **SCIF is set**
- **Software write to I2CxB TXB clears CSTR**

**R/W**
- **R/W copied from matching address**
- **ADRF is set**

**D/A**
- **matching address copy to I2CxA DB0**

**ACKSTAT**
- **Master’s ACK copied to ACKSTAT**
- **NACKIF is set**
- **ACKTIF is set**

**i2C BUS**
- **Slave transmitting data**
- **SCL and SDA signals**

**TXIF NOT set**
- **No new TX data on I2CxCNT = 0**
- **(shifter loaded 8'b1111 1111)**
FIGURE 33-10: I2C SLAVE, 7-BIT ADDRESS, TRANSMISSION (NO CLOCK STRETCHING)
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 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.

33.4.3.4 Slave Reception (10-bit Addressing Mode)

This section describes the sequence of events for the I²C module configured as an I²C 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 R/W = 0.
3. The received high address is compared with the values in I2CxADR1 and I2CxADR3 registers.
4. If high address matches; R/W is copied to R/W bit, D/A bit is cleared, high address data is copied to I2CxADB0, and ACTDT is copied to SDA. 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 I2CxADB0 and I2CxADB1 registers and change the value of ACKDT before releasing SCL by clearing CSTR.
7. Master sends ninth SCL pulse for ACK.
8. If Data write interrupt and hold is enabled (WRIE = 1), CSTR is set, I2CxIF is set. Slave software can read data from I2CxRXB to resume communication.
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 ACTDT 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 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/A 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 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.

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.
I2C SLAVE, 10-BIT ADDRESS, RECEPTION WITH STOP (ADB = 1)

From Slave to Master
- SDA set only after full address match
- SMA set only after full address match
- Hardware clears SMA
- Software reads address from I2CxRXB, clearing I2CxRXIF
- Software reads data from I2CxRXB, clearing I2CxRXIF

Receive Data
- Receive Low Address
- Receive High Address

Acknowledgment
- ACKTIF = 0 for high address /w
- ACKTIF = set
- ACKTIE = 0
- ADRIE = 0
- ADRIF is set
- PCIF is set
- SMA copied from matching address R/W value copied

Event Flags
- SCIF is set
- CSTR
- I2CxRXIF is set
- RXBF

DA

High Address copied to I2CxRXB
- R/W = 0 from I2CAADB [10]

Low Address copied to I2CxRXB
- R/W copied from matching address

ADC

Stop
- R/W = 0 from I2CAADB [10]
33.4.3.5 Slave Transmission (10-bit Addressing Mode)

This section describes the sequence of events for the I2C module configured as an I2C 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 I2CxFIR register is set. If Start condition interrupt is enabled (SCIE bit is set), generic interrupt I2CxFIF is set.
2. Master transmits high address byte with R/W = 0.
3. The received high address is compared with the values in I2CxADR1 and I2CxADR3 registers.
4. If high address matches; R/W is copied to R/W bit, D/A 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. I2CxFIF 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 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 ACTDT 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 ACK.
14. If the Acknowledge interrupt and hold is enabled (ACKTIE = 1), CSTR is set, I2CxFIF 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 I2CxFIF is set.
17. Master transmits high address byte with R/W = 1.

18. If SMA = 1, and if high address matches; R/W is copied to R/W bit, D/A bit is cleared, high address data is copied to I2CxADB1, and ACTDT is output to SDA. If the address does not match or SMA = 0; module become idle.
19. If ADRIE = 1, CSTR is set. I2CxFIF 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 I2CCNT!= 0 (I2CTXIF = 1), CSTR is set. Slave software must load data into I2CxADB0 to release SCL.
21. Master sends SCL pulse for ACK. If I2CCNT = 0, CNTIF is set.
22. If NACK; NACKIF is set, slave goes idle.
23. If ACKTIE = 1, CSTR is set, I2CxFIF 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.
FIGURE 33-12: \textit{I}^2\textit{C} SLAVE, 10-BIT ADDRESS, TRANSMISSION
33.5 **I\(^2\)C Master Mode**

Master mode is enabled by setting and clearing the appropriate Mode<2:0> bits in I2CxCON and then by setting the I2CEN bit. Master mode of operation is supported by interrupt generation on buffer full (RXIF), buffer empty (TXIF), and the detection of the Start, Restart, and Stop conditions. The Stop (P), Restart (RS) and Start (S) bits are cleared from a Reset or when the I\(^2\)C module is disabled. Control of the I\(^2\)C bus is asserted when the BFRE bit of I2CSTAT0 is set.

33.5.1 **I\(^2\)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\(^2\)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 I2CxADB0/1 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\(^2\)C module clock source is selected by the I2CxCCLK register. The I\(^2\)C Clock provides the SCL output clock for Master mode and is used by the Bus Free timer. The I\(^2\)C clock can be sourced from several peripherals.

33.5.3 **Bus Free Time**

In Master modes, the BFRE bit of the I2CSTAT0 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\(^2\)C clock input before asserting the BFRE bit. The BFRET bits are used to ensure that the I\(^2\)C module always follows the minimum Stop Hold Time. The I\(^2\)C timing requirements are listed in the electrical specifications chapter.

Note: I\(^2\)C clock is not required to have a 50% duty cycle.

33.5.4 **Master Clock Timing**

The clock generation in the I\(^2\)C module can be configured using the Fast Mode Enable (FME) bit of the I2CxADB2 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\(^2\)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\(^2\)C clock, proceeding only after detecting SCL high. **Figure 33-13** shows the clock synthesis timing when FME = 0.
33.5.4.2 Clock Timing with FME = 1

One TSCL, consists of four clocks of the \( \text{I}^2\text{C} \) 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 \( \text{I}^2\text{C} \) 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 I²C 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 TSCL 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 I²C 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 Tscl/2. Then the SCL pin is released floated high) for Tscl/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 Tscl. 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.
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 (TSCl) and the SCL pin is released high. When the SCL pin is sampled high (clock arbitration), the master counts another TSCl. The SCL pin is then pulled low. Figure 33-17 shows the timings for Acknowledge sequence.

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 TSCl/2 later and is detected high. The SDA pin is then released. When the SDA pin transitions high while SCL is high, the PCIF bit of the I2CxIF register is set. Figure 33-18 shows the timings for a Stop condition.
33.5.9 MASTER TRANSMISSION IN 7-BIT ADDRESSING MODE

This section describes the sequence of events for the I²C module configured as an I²C 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.

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.

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 I2CTXB to the shift register. The value of I2CxCNT is decremented.

FIGURE 33-19:  \(^2\)C MASTER, 7-BIT ADDRESS, TRANSMISSION WITH STOP

- **SDA** and **SCL** signals:
  - A7, A6, A5, A4, A3, A2, A1
  - ACK, D7, D6, D5, D4, D3, D2, D1, D0

- **MMA**:
  - Software sets START to start transmission
  - SCIF is set

- **ACKSTAT**:
  - R/W = 0 from I2CxADB[0]
  - Address copied from I2CxADB[7:1]

- **I2CxCNT**:
  - 0x02, 0x01, 0x00

- **TXBE**:
  - Before Start, software loads one byte in I2CxTXB
  - MSb is of I2CxTXB copied to SDA
  - Data byte loaded from I2CxTXB to shifter
  - I2CxTXIF is set
  - Software writes
  - I2CxTXB
  - MSb is of I2CxTXB copied to SDA
  - Second data byte loaded from I2CxTXB to shifter
  - I2CxTXIF NOT set
  - No new TX data on I2CxCNT = 0
    - (shifter loaded 8'b11111111)

- **From Slave to Master**:
  - MMA
  - SCIF is set
  - PCIF is set

- **From Master Transmitting Data**:
  - ACK
  - Data byte loaded from I2CxTXB to shifter
  - I2CxTXIF NOT set
  - No new TX data on I2CxCNT = 0
  - (shifter loaded 8'b11111111)
33.5.10 MASTER RECEPTION IN 7-BIT ADDRESSING MODE

This section describes the sequence of events for the I\(^2\)C module configured as an I\(^2\)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 = d 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 9th 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 7th 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.
FIGURE 33-20: \( \text{i}^2\text{C} \) MASTER, 7-BIT ADDRESS, RECEPTION

- Slave Sends ACK after \( R/W = 1 \) from I2CxADB1[0]
- Master sends ACK
- Master sends NACK
- Slave Transmits Data
- Slave Transmitting Data
- Slave Address copied from I2CxADB1[7:1]
- Software sets START to start transmission
- Hardware clears MMA
- SCIF is set
- Master's ACK copied from ACKDT
- NACK on CNT = 0
- NACKIF is set
- PCIF is set
- MDR
- I2CxRXF is set
- I2CxRXIF is set
- Software reads data from I2CxRXB
- Software reads I2CxRXB
- RXBF
- ACKDT
- I2CxCNT
- \( \text{0x02} \)
- \( \text{0x02} \)
- \( \text{0x01} \)
- \( \text{0x00} \)
- MMA
- SDA
- SCL
- RSEN = 0; Master sends stop condition

1. SlaveTransmittingData
2. SlaveTransmittingData
3. \( \text{A7} \)
4. \( \text{A6} \)
5. \( \text{A5} \)
6. \( \text{A4} \)
7. \( \text{A3} \)
8. \( \text{A2} \)
9. \( \text{A1} \)
10. \( \text{D7} \)
11. \( \text{D6} \)
12. \( \text{D5} \)
13. \( \text{D4} \)
14. \( \text{D3} \)
15. \( \text{D2} \)
16. \( \text{D1} \)
17. \( \text{D0} \)
18. \( \text{S} \)
19. \( \text{P} \)
33.5.11 MASTER TRANSMISSION IN 10-BIT ADDRESSING MODE

This section describes the sequence of events for the I²C module configured as an I²C 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/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/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 ACK, master hardware sends the low address byte from I2CxADB0.

   If ABD = 1; i.e., Address buffer are disabled
   If 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! = 0, I2CxTXIF and MDR bits are set. Clock is stretched on 8th falling SCL edge until master software writes next data byte to I2CxTXB.

6. Master hardware sends ninth SCL pulse for 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 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 ACK and CNTIF is set.

9. If I2CxCNT! = 0; go to step 5.
FIGURE 33-21: I2C MASTER, 10-BIT ADDRESS, TRANSMISSION WITH STOP

Transmitting Data

From Slave to Master

ACK

ACK

Transmitting Data

I2CxCNT = 0

Master sends Stop

ACKSTAT

High Address copied from I2CxADB1[7:1]

Low Address copied from I2CxADB0[7:0]

Data byte loaded from I2CxTXB to shifter

I2CxTXIF NOT set

No new TX data on I2CxCNT = 0

(shifter loaded 8'b1111 1111)

Before Start, software loads one byte in I2CxTXB

I2CxTXIF NOT set

Software sets START to start transmission

SCIF is set

Hardware clears MMA

PCIF is set

From Slave to Master

ACK

Rev. 10-000 302A
11/2/2016

SDA
1 1 1 1 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0 ACK D7 D6 D5 D4 D3 D2 D1 D0

SCL
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9

MMA

R/W = 0 from I2CxADB1[0]

Hardware clears MMA

PCIF is set

I2CxCNT = 0

Master sends Stop

0x01

0x01

0x00

0x00

I2CxTXIF

I2CxADB1[7:1]

I2CxADB0[7:0]

0x01

0x01
33.5.12 MASTER RECEPTION IN 10-BIT ADDRESSING MODE

This section describes the sequence of events for the \( I^2C \) module configured as an \( I^2C \) master in 10-bit Addressing mode and is receiving data. Figure 33-22 is used as a visual reference for this description.

1. Master software loads high address byte in \( I^2CxADB1 \) and low address byte in \( I^2CxADB0 \) for write and sets restart enable (RSTEN) bit.
2. Master software sets START bit.
3. Master hardware waits for BFRE bit to be set; then shifts out start, high address and waits for acknowledge.
4. If slave responds with a NACK, master hardware sends Stop and ends communication.
5. If slave responds with ACK, master hardware shifts out the low address.
6. If the transmit buffer empty flag (TXBE) is set and \( I^2CxCNT! = 0 \), the clock is stretched on 8th falling SCL edge. Allowing master software writes next data to \( I^2CxTXB \).
7. Master hardware sends 9th SCL pulse for ACK from slave and loads the shift register from \( I^2CxTXB \).
8. If slave responds with a NACK, master hardware sends Stop and ends communication.
9. If slave responds with an ACK and \( I^2CxCNT = 0 \), master hardware sets MDR bit, go to Step 11.
10. If slave responds with an ACK and \( I^2CxCNT! = 0 \), master hardware outputs data in shift register on SDA and waits for ACK from slave. Go to step 4.
11. Master software loads \( I^2CxADB0 \) for read, and \( I^2CCNT \) with the number of bytes to be received in the current transaction.
12. Master software sets Start bit.
13. Master hardware shifts out Restart and high address with \( R/W = 1 \).
14. Master sends out 9th SCL pulse for ACK from Slave.
15. If slave responds with a NACK, master hardware sends Stop or sets MDR (RSEN bit).
16. If slave responds with an ACK, master hardware shifts 7 bits of data into the shift register from the slave.
17. If the receive buffer full flag (RXBF) is set, clock is stretched on seventh falling SCL edge.
18. Master software can clear clock stretching by reading the previous data in the receive buffer.
19. Master hardware shifts 8th bit of data into the shift register from slave and loads it into \( I^2CxRXB \).
20. Master software reads data from \( I^2CxRXB \) register.

21. If \( I^2CxCNT! = 0 \), master hardware clocks out ACKDT as ACK value to slave.
22. If \( I^2CxCNT = 0 \), master hardware clocks out ACKCNT as ACK value to slave
23. Go to step 4.
FIGURE 33-22: I2C MASTER, 10-BIT ADDRESS, RECEPTION (USING RSTEN BIT)

- **SDA**
  - Master sends start event
  - R/W = 0 from I2CxADB1[0]
  - SCL
  - High Address copied from I2CxADB1[7:1]
  - Low Address copied from I2CxADB0[7:0]
  - Software sets START to start transmission

- **MMA**
  - Software sets RSEN before setting START
  - Hardware clears MMA

- **MDR**
  - MDR cleared by setting START
  - Software clears RSEN before setting START

- **ACKSTAT**
  - Software sets RSEN before setting START

- **RSEN**
  - Software sets RSEN before setting START

- **I2CxCNT**
  - 0x00
  - 0x00
  - 0x01
  - 0x00
  - Software writes I2CxCNT before setting START

- **RXBF**
  - Master sends Stop
  - I2CxRXIF is set
33.6 I²C 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 I²C 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. MSTACK 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 I²C 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.
FIGURE 33-24: I2C MULTI-MASTER, 7-BIT ADDRESS, WRITE (ADRIE = 1, WRIE = 0)

- Address copied from I2CxADB1
- Another Master clocks ACK and begins sending data
- Received Data
- Received Data
- Other Master sends stop condition

SDA
A7 A6 A5 A4 A3 A2 A1
S
\[\text{ACK}\]
D7 D6 D5 D4 D3 D2 D1 D0
\[\text{ACK}\]
CSTR
ADRIF is set
CSTR cleared by software
MMA
Software sets START
Master loses arbitration of address
BCLIF is set, hardware clears MMA
Continues to clock in slave address
\[\text{ACKDT}\]
ACKDT cleared by software
User must clear BCLIF to send ACK
I2CxCNT
Software updates I2CxCNT for Slave receive message
0x07 0x02 0x01 0x00
\[\text{RXBF}\]
I2CxRXIF is set
Software reads data from I2CXRxB
Software reads I2CXRxB
RXBF
33.7 Register Definitions: I\(^2\)C Control

This section defines all the registers associated with the control and status of the I\(^2\)C bus.

**REGISTER 33-1: I2CxCON0: I\(^2\)C CONTROL REGISTER 0**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td><strong>EN</strong>: I(^2)C Module Enable bit</td>
</tr>
<tr>
<td></td>
<td>1 = Enables the I(^2)C module (^{(1,2)})</td>
</tr>
<tr>
<td></td>
<td>0 = Disables the I(^2)C module</td>
</tr>
<tr>
<td>6</td>
<td><strong>RSEN</strong>: Restart Enable bit (Only mode&lt;2:0&gt; = 1xx)</td>
</tr>
<tr>
<td></td>
<td>1 = When (I2CCNT = 0 or ACKSTAT = 1), on 9th falling SCL sets MDR.</td>
</tr>
<tr>
<td></td>
<td>0 = When (I2CCNT = 0 or ACKSTAT = 1), on 9th falling SCL: master shifts out a Stop condition</td>
</tr>
<tr>
<td>5</td>
<td><strong>S</strong>: Master Start/Restart bit (Only Mode&lt;2:0&gt; = 1xx)</td>
</tr>
<tr>
<td></td>
<td>When MMA = 0</td>
</tr>
<tr>
<td></td>
<td>1 = Set by user set of START bit or write to I2CTXB, waits for BFRE = 1 to begin with a Start</td>
</tr>
<tr>
<td></td>
<td>0 = Cleared by hardware after sending Start</td>
</tr>
<tr>
<td></td>
<td>When (MMA = 1 &amp; MDR = 1 &amp; pause_for_Restart)</td>
</tr>
<tr>
<td></td>
<td>1 = Set by user set of START bit or write to I2CTXB, resumes communication with a Restart</td>
</tr>
<tr>
<td></td>
<td>0 = Cleared by hardware after sending Restart</td>
</tr>
<tr>
<td></td>
<td>Else - Writes to I2CTXB or set has no effect on Start bit</td>
</tr>
<tr>
<td>4</td>
<td><strong>CSTR</strong>: Slave Clock Stretching bit (^{(3)})</td>
</tr>
<tr>
<td></td>
<td>1 = Clock is held low (clock stretching)</td>
</tr>
<tr>
<td></td>
<td>0 = Enable clocking, SCL control is released</td>
</tr>
<tr>
<td></td>
<td><strong>SMA = 1 and RXBF = 1</strong> (^{(6)})</td>
</tr>
<tr>
<td></td>
<td>- Set by hardware on 7th falling SCL edge</td>
</tr>
<tr>
<td></td>
<td>- User must read byte I2CRXB to release SCL</td>
</tr>
<tr>
<td></td>
<td><strong>SMA = 1 and TXBE = 1 and I2CCNT= 0</strong> (^{(5)})</td>
</tr>
<tr>
<td></td>
<td>- Set by hardware on 8th falling SCL edge</td>
</tr>
<tr>
<td></td>
<td>- User must write byte to I2CTXB to release SCL</td>
</tr>
<tr>
<td></td>
<td>when ADRIE is set (^{(4)})</td>
</tr>
<tr>
<td></td>
<td>- Set by hardware on 8th falling SCL edge of matching received address</td>
</tr>
<tr>
<td></td>
<td>- User must clear CSTR to release SCL</td>
</tr>
<tr>
<td></td>
<td><strong>SMA = 1 &amp; WRIE = 1</strong> (^{(1)})</td>
</tr>
<tr>
<td></td>
<td>- Set by hardware on 8th falling SCL edge of received data byte</td>
</tr>
<tr>
<td></td>
<td>- User must clear CSTR to release SCL</td>
</tr>
<tr>
<td></td>
<td><strong>SMA = 1 &amp; ACKTIE = 1</strong> (^{(1)})</td>
</tr>
<tr>
<td></td>
<td>- Set by hardware on 9th falling SCL edge</td>
</tr>
<tr>
<td></td>
<td>- User must clear CSTR to release SCL</td>
</tr>
</tbody>
</table>

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 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 I2CRXB to release SCL
- **MMA = 1 & TXBE = 1 & I2CCNT!= 0**
  - **pause_for_tx** - Set by hardware on 8th falling SCL edge
  - User must write to I2CTXB to release SCL
- **ADB = 1**
  - I2CCNT is ignored for the high and low address in 10-bit mode
- **pause_for_restart** - Set by hardware on 9th falling SCL edge
  - RSEN = 1 & MMA = 1 & I2CCNT = 0 || ACKSTAT = 1
  - User must set START or write to I2CTXB to release SCL and shift Restart onto bus

bit 2-0  
**MODE<2:0>:** I2C Mode Select bits

- **111** = I2C Multi-Master mode (SMBus 2.0 Host), *(f)*
  - Works as both mode<2:0> = 001 and mode<2:0> = 100
- **110** = I2C Multi-Master mode (SMBus 2.0 Host), *(f)*
  - Works as both mode<2:0> = 000 and mode<2:0> = 100
- **101** = I2C Master mode, 10-bit address
- **100** = I2C Master mode, 7-bit address
- **011** = I2C Slave mode, one 10-bit address with masking
- **010** = I2C Slave mode, two 10-bit address
- **001** = I2C Slave mode, two 7-bit address with masking
- **000** = I2C 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 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: I2C CONTROL REGISTER 1

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>ACKCNT: Acknowledge End of Count bit(2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Acknowledge value transmitted after received data, when I2CCNT = 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = Not Acknowledge (copied to SDA output)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = Acknowledge (copied to SDA output)</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>ACKDT: Acknowledge Data bit(1,2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Acknowledge value transmitted after matching address</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Acknowledge value transmitted after received data, when I2CCNT! = 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = Not Acknowledge (copied to SDA output)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = Acknowledge (copied to SDA output)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>ACKSTAT: Acknowledge Status bit (Transmission only)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = Acknowledge was not received for most recent transmission</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = Acknowledge was received for most recent transmission</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>ACKT: Acknowledge Time Status bit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = Indicates the I2C bus is in an Acknowledge sequence, set on 8th falling edge of SCL clock</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = Not in Acknowledge sequence, cleared on 9th rising edge of SCL</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Unimplemented: Read as ‘1’b0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>RXO: Receive Overflow Status bit (MODE&lt;2:0&gt; = 0xx &amp; 11x)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This bit can only be set when CSD= 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = Set when SMA = 1, and a master clocks in data when RXBF = 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = No slave overflow condition</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>TXU: Transmit Underflow Status bit (MODE&lt;2:0&gt; = 0xx &amp; 11x)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This bit can only be set when CSTRDIS = 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = Set when SMA = 1, and a master clocks out data when TXBE = 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = No slave underflow condition</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>CSD: Clock Stretching Disable bit (MODE&lt;2:0&gt; = 0xx &amp; 11x)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = When SMA = 1, the CSTR bit will never be set</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = Slave clock stretching proceeds normally</td>
<td></td>
</tr>
</tbody>
</table>

**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 I2C hardware when bus errors are indicated in the I2CxSTAT1 or I2CxERR registers.
### REGISTER 33-3: I2CxCON2: I2C CONTROL REGISTER 2

<table>
<thead>
<tr>
<th>Bit</th>
<th>ACNT</th>
<th>GCEN</th>
<th>FME</th>
<th>ADB</th>
<th>SDAHT&lt;1:0&gt;</th>
<th>BFRET&lt;1:0&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
</tbody>
</table>

#### 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 I2C Count Register Enable bit

- **1**: The first received or transmitted byte after the address, is automatically loaded into the I2CCNT register. The I2CCNT 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 I2CCNT<NACK> from being sent for the byte that would update the I2CCNT register.
- **0**: Auto-load of I2CCNT 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 = FCLK/4)
- **0**: SCL is sampled high twice before driving SCL low. (FSCL = FCLK/5)

**bit 4**: **ADB**: Address Data Buffer Disable bit

- **1**: Received address data is loaded into both the I2CADD and I2CRXB
  - Transmitted address data is loaded from the I2CTXB
- **0**: Received address data is loaded only into the I2CADD
  - Transmitted address data is loaded from the I2CADD0/1 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 I2C Clock pulses
- **10**: 32 I2C Clock pulses
- **01**: 16 I2C Clock pulses
- **00**: 8 I2C Clock pulses
REGISTER 33-4: I2CxCLOCK: I²C CLOCK SELECTION REGISTER

<table>
<thead>
<tr>
<th>bit 7-4</th>
<th>bit 3-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unimplemented: Read as '0'</td>
<td></td>
</tr>
<tr>
<td>CLK&lt;3:0&gt;: I²C Clock Selection Bits</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLK&lt;3:0&gt;</th>
<th>I²C Clock Selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>1010-1111</td>
<td>Reserved</td>
</tr>
<tr>
<td>1001</td>
<td>SMT1 overflow</td>
</tr>
<tr>
<td>1000</td>
<td>TMR6 post scaled output</td>
</tr>
<tr>
<td>0111</td>
<td>TMR4 post scaled output</td>
</tr>
<tr>
<td>0110</td>
<td>TMR2 post scaled output</td>
</tr>
<tr>
<td>0101</td>
<td>TMR0 overflow</td>
</tr>
<tr>
<td>0100</td>
<td>Clock Reference output</td>
</tr>
<tr>
<td>0011</td>
<td>MFINTOSC (500 kHz)</td>
</tr>
<tr>
<td>0010</td>
<td>HFINTOSC</td>
</tr>
<tr>
<td>0001</td>
<td>FOSC</td>
</tr>
<tr>
<td>0000</td>
<td>FOSC/4</td>
</tr>
</tbody>
</table>

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
REGISTER 33-5:  I2CxBTO: I2C BUS TIMEOUT SELECTION REGISTER

| bit 7-3 | Unimplemented: Read as ‘0’ |
| bit 2-0 | BTO<2:0>: I2C Bus Timeout Selection bits |

<table>
<thead>
<tr>
<th>BTO&lt;2:0&gt;</th>
<th>I2Cx Bus Timeout Selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>CLC4OUT</td>
</tr>
<tr>
<td>110</td>
<td>CLC3OUT</td>
</tr>
<tr>
<td>101</td>
<td>CLC2OUT</td>
</tr>
<tr>
<td>100</td>
<td>CLC1OUT</td>
</tr>
<tr>
<td>011</td>
<td>TMR6 post scaled output</td>
</tr>
<tr>
<td>010</td>
<td>TMR4 post scaled output</td>
</tr>
<tr>
<td>001</td>
<td>TMR2 post scaled output</td>
</tr>
<tr>
<td>000</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

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
REGISTER 33-6: I2CxSTAT0: I²C STATUS REGISTER 0

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>BFRE(3)</td>
<td>SMA</td>
<td>MMA</td>
<td>R(1, 2)</td>
<td>D</td>
<td>—</td>
</tr>
</tbody>
</table>

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(3)
1 = Indicates the I²C bus is idle  
Both SCL and SDA have been high for time-out selected by I2CCON2<BFRET<1:0>> bits.
I2CCCLK must select a valid clock source for this bit to function.
0 = Bus not idle (When no I2CCLK 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(1, 2)
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: I2C STATUS REGISTER 1

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value at POR/BOB</th>
<th>Value at all other Resets</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td><strong>TXWE</strong> (2) Transmit Write Error Status bit (2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = A new byte of data was written to I2CTXB when it was full (Must be cleared by software)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = No transmit write error</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td><strong>TXBE</strong> Transmit Buffer Empty Status bit</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = I2CTXB is empty (Cleared by writing the I2CTXB register)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = I2CTXB is full</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td><strong>RXRE</strong> Receive Read Error Status bit</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = A byte of data was read from I2CRXB when it was empty. (Must be cleared by software)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = No receive overflow</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td><strong>CLRBF</strong> Clear Buffer bit</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Setting this bit clears/empties the receive and transmit buffers, causing reset of RXBF and TXBE.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Setting this bit clears the RXIF and TXIF interrupt flags.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>This bit is set-only special function, and always reads '0'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td><strong>RXBF</strong> Receive Buffer Full Status bit</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = I2CRXB has received new data ( Cleared by reading the I2CRXB register)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = I2CRXB is empty</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td><strong>RXBE</strong> Transmit Buffer Empty Status bit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td><strong>RXRE</strong> Receive Read Error Status bit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td><strong>RXBF</strong> Receive Buffer Full Status bit</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as '0'
- **u** = Bit is unchanged
- **x** = Bit is unknown
- **’1’** = Bit is set
- **’0’** = Bit is cleared
- **HS** = Hardware set
- **HC** = Hardware clear

**Note:**
1: The bits are held in Reset when I2CEN = 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: I²C ERROR REGISTER

<table>
<thead>
<tr>
<th>U-0</th>
<th>R/W/HS-0</th>
<th>R/W/HS-0</th>
<th>R/W/HS-0</th>
<th>U-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>BTOIF(1,2)</td>
<td>BCLIF(1)</td>
<td>NACKIF(1)</td>
<td>—</td>
<td>BTOIE</td>
<td>BCLIE</td>
<td>NACKIE</td>
</tr>
</tbody>
</table>

**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 Time-Out Interrupt Flag bit(1,2)
- 1 = Bus Timeout occurred
- 0 = No bus timeout

**bit 5**  
**BCLIF:** Bus Collision Detect Interrupt Flag bit(1)
- 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(1)
- 1 = When (SMA = 1 || MMA = 1) and a NACK is detected on the bus
  - NACKIF is also set when any of the TXWRE, RXRDE, TXUF, RXOVR 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 Time-Out Interrupt Enable bit
- 1 = Enable interrupt on bus time out
- 0 = Bus Tim-out 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<I2CEIF> bit.

**2:** User software must select the Bus Time-out Source in the I2CBTO register.
### REGISTER 33-9: I2CxCNT: I2C BYTE COUNT REGISTER

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Readable</th>
<th>Writable</th>
<th>Unimplemented</th>
<th>Value at POR and BOR/Value at all other Resets</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>CNT&lt;7&gt;</td>
<td>R</td>
<td>W</td>
<td>U</td>
<td>-n = Value at POR and BOR</td>
</tr>
<tr>
<td>0</td>
<td>CNT&lt;0&gt;</td>
<td>R</td>
<td>W</td>
<td>U</td>
<td>-n = Value at POR and BOR</td>
</tr>
</tbody>
</table>

**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>: I2C 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

<table>
<thead>
<tr>
<th>R/W/HS-0</th>
<th>R/W/HS-0</th>
<th>U-0</th>
<th>R/W/HS-0</th>
<th>R/W/HS-0</th>
<th>R/W/HS-0</th>
<th>R/W/HS-0</th>
<th>R/W/HS-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTIF</td>
<td>ACKTIF</td>
<td>—</td>
<td>WRIF</td>
<td>ADRIF</td>
<td>PCIF</td>
<td>RSCIF</td>
<td>SCIF</td>
</tr>
</tbody>
</table>

**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

**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 I2CCNT = 0, set by the 9th falling edge of SCL.
  - **0** = I2CCNT condition has not occurred.

**bit 6**
- **ACKTIF**: Acknowledge Status Time Interrupt Flag bit (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.

**Note 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

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 6</th>
<th>bit 5</th>
<th>bit 4</th>
<th>bit 3</th>
<th>bit 2</th>
<th>bit 1</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTIE</td>
<td>ACKTIE</td>
<td>U-0</td>
<td>WRIE</td>
<td>ADRIE</td>
<td>PCIE</td>
<td>RSCIE</td>
<td>SCIE</td>
</tr>
</tbody>
</table>

#### 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 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
- 1 = Enable interrupt on detection of Stop condition
- 0 = Stop detection interrupts are disabled

#### bit 1  **RSCIE**: Restart Condition Interrupt Enable
- 1 = Enable interrupt on detection of Restart condition
- 0 = Start detection interrupts are disabled

#### bit 0  **SCIE**: Start Condition Interrupt Enable
- 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<I2CxlF> bit.
## REGISTER 33-12: I2CxADR0: I\(^2\)C ADDRESS 0 REGISTER

<table>
<thead>
<tr>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADR7</td>
<td>ADR6</td>
<td>ADR5</td>
<td>ADR4</td>
<td>ADR3</td>
<td>ADR2</td>
<td>ADR1</td>
<td>ADR0</td>
</tr>
</tbody>
</table>

### 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 1 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: I2C ADDRESS 1 REGISTER

<table>
<thead>
<tr>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>U-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADR14</td>
<td>ADR13</td>
<td>ADR12</td>
<td>ADR11</td>
<td>ADR10</td>
<td>ADR9</td>
<td>ADR8</td>
</tr>
<tr>
<td>bit 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>U-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADR7</td>
<td>ADR6</td>
<td>ADR5</td>
<td>ADR4</td>
<td>ADR3</td>
<td>ADR2</td>
<td>ADR1</td>
</tr>
<tr>
<td>bit 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as ‘0’
- **u** = Bit is unchanged
- **x** = Bit is unknown
- **‘1’** = Bit is set
- **‘0’** = Bit is cleared
- **HS** = Hardware set
- **HC** = Hardware clear

**bit 7-1**

**ADR[7-1]:** Address or Divider bits

- **MODE<2:0> = 000** | **110** - 7-bit Slave/Multi-Master Modes
  - **ADR<7:1>:** 7-bit Slave Address
  - **ADR<0>:** Unused in this mode; bit state is a don’t care

- **MODE<2:0> = 001** | **111** - 7-bit Slave/Multi-Master modes w/Masking
  - **MSK0<7:1>:** 7-bit Slave Address
  - **MSK0<0>:** Unused in this mode; bit state is a don’t care

- **MODE<2:0> = 01x** - 10-bit Slave Modes
  - **ADR<14-10>:** Bit pattern sent by master is fixed by I2C 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: I2C ADDRESS 2 REGISTER

<table>
<thead>
<tr>
<th>bit 7-0</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADR7</td>
<td>ADR6</td>
<td>ADR5</td>
<td>ADR4</td>
<td>ADR3</td>
<td>ADR2</td>
<td>ADR1</td>
<td>ADR0</td>
<td></td>
</tr>
</tbody>
</table>

**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
    - **MSK0<7-0>:** The received address byte is masked, then compared to I2CxADR0
REGISTER 33-15: I2CXADR3: I2C ADDRESS 3 REGISTER

<table>
<thead>
<tr>
<th>bit 15</th>
<th>bit 8</th>
<th>bit 7</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADR14</td>
<td>ADR13</td>
<td>ADR12</td>
<td>ADR11</td>
</tr>
<tr>
<td>ADR10</td>
<td>ADR9</td>
<td>ADR8</td>
<td>U-0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADR7</td>
<td>ADR6</td>
</tr>
<tr>
<td>ADR5</td>
<td>ADR4</td>
</tr>
<tr>
<td>ADR3</td>
<td>ADR2</td>
</tr>
<tr>
<td>ADR1</td>
<td>U-0</td>
</tr>
</tbody>
</table>

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 3 bits

**MODE<2:0> = 000** - 7-bit Slave/Multi-Master Modes

**ADR<7:1>:** 7-bit Slave Address

**ADR<0>:** Unused in this mode; bit state is a don’t care

**MODE<2:0> = 001** - 7-bit Slave/Multi-Master Mode with Masking

**MSK1<7:1>:** 7-bit Slave Address

**MSK1<0>:** Unused in this mode; bit state is a don’t care

**MODE<2:0> = 010** - 10-Bit Slave Mode

**ADR<14-10>:** Bit pattern sent by master is fixed by I2C 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

**MSK0<14-8>:** The received address byte, bit n, is compared to I2CxADR0 to detect I2C address match
### REGISTER 33-16: I2CxADB0: I2C ADDRESS DATA BUFFER 0 REGISTER(1)

<table>
<thead>
<tr>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADB7</td>
<td>ADB6</td>
<td>ADB5</td>
<td>ADB4</td>
<td>ADB3</td>
<td>ADB2</td>
<td>ADB1</td>
<td>ADB0</td>
</tr>
</tbody>
</table>

#### bit 7-0 MODE<2:0> = 00x
- **ADB<7:1>:** Address Data byte
- **R/W:** Read/not-Write Data bit

#### bit 7-0 MODE<2:0> = 01x
- **ADB<7:0>:** Address Data byte
- **R/W:** Read/not-Write Data bit

#### bit 7-0 MODE<2:0> = 100
- **ADB<7:0>:** Address Data byte
- **R/W:** Read/not-Write Data bit

#### bit 7-0 MODE<2:0> = 101
- **ADB<7:0>:** Low Address Data byte
- **R/W:** Read/not-Write Data bit

#### bit 7-0 MODE<2:0> = 11x
- **ADB<7:1>:** Address Data byte
- **R/W:** Read/not-Write Data bit

---

**Note 1:** This register is read only except in master, 10-bit Address mode (MODE<2:0> = 101).
REGISTER 33-17: I2CxADB1: I2C ADDRESS DATA BUFFER 1 REGISTER(1)

<table>
<thead>
<tr>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADB7</td>
<td>ADB6</td>
<td>ADB5</td>
<td>ADB4</td>
<td>ADB3</td>
<td>ADB2</td>
<td>ADB1</td>
<td>ADB0</td>
</tr>
</tbody>
</table>

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
Master hardware uses this bit to produce read versus write operations.

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
Master hardware uses this bit to produce read versus write operations.

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
Master hardware uses this bit to produce read versus write operations

Note 1:  This register is read only in slave, 7-bit Addressing modes (MODE<2:0> = 0xx)
### TABLE 33-18: SUMMARY OF REGISTERS FOR I2C 8-BIT MACRO

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2CxBTO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>582</td>
</tr>
<tr>
<td>I2CxCLK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>581</td>
</tr>
<tr>
<td>I2CxPIE</td>
<td>CNTIE</td>
<td>ACKTIE</td>
<td></td>
<td>WRIE</td>
<td>ADRIE</td>
<td>PCIE</td>
<td>RSCIE</td>
<td>SCIE</td>
<td>588</td>
</tr>
<tr>
<td>I2CxPIR</td>
<td>CNTIF</td>
<td>ACKTIF</td>
<td></td>
<td>WRIF</td>
<td>ADRIF</td>
<td>PCIF</td>
<td>RSCIF</td>
<td>SCIF</td>
<td>587</td>
</tr>
<tr>
<td>I2CxERR</td>
<td></td>
<td></td>
<td></td>
<td>BTOIF</td>
<td>BCLIF</td>
<td>NACKIF</td>
<td></td>
<td></td>
<td>585</td>
</tr>
<tr>
<td>I2CxSTAT0</td>
<td>BFORE</td>
<td>SMA</td>
<td>MMA</td>
<td>R</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td>583</td>
</tr>
<tr>
<td>I2CxSTAT1</td>
<td>TXWE</td>
<td>TXBE</td>
<td></td>
<td>RXRE</td>
<td>CLRBF</td>
<td></td>
<td></td>
<td></td>
<td>584</td>
</tr>
<tr>
<td>I2CxCON0</td>
<td>EN</td>
<td>RSEN</td>
<td>S</td>
<td>CSTR</td>
<td>MDR</td>
<td>MODE&lt;2:0&gt;</td>
<td></td>
<td></td>
<td>577</td>
</tr>
<tr>
<td>I2CxCON1</td>
<td>ACKCNT</td>
<td>ACKDT</td>
<td>ACKSTAT</td>
<td>ACKT</td>
<td></td>
<td>RXOV</td>
<td>TXU</td>
<td>CSD</td>
<td>579</td>
</tr>
<tr>
<td>I2CxCON2</td>
<td>ACNT</td>
<td>GCEN</td>
<td>FME</td>
<td>ADB</td>
<td>SDAHT&lt;3:2&gt;</td>
<td>BFRET&lt;1:0&gt;</td>
<td></td>
<td></td>
<td>580</td>
</tr>
<tr>
<td>I2CxADR0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>589</td>
</tr>
<tr>
<td>I2CxADR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>590</td>
</tr>
<tr>
<td>I2CxADR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>591</td>
</tr>
<tr>
<td>I2CxADR3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>592</td>
</tr>
<tr>
<td>I2CxADB0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>593</td>
</tr>
<tr>
<td>I2CxADB1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>594</td>
</tr>
<tr>
<td>I2CxCNT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>586</td>
</tr>
<tr>
<td>I2CxPIR</td>
<td>CNTIF</td>
<td>ACKTIF</td>
<td></td>
<td>WRIF</td>
<td>ADRIF</td>
<td>PCIF</td>
<td>RSCIF</td>
<td>SCIF</td>
<td>587</td>
</tr>
<tr>
<td>I2CxPIE</td>
<td>CNTIE</td>
<td>ACKTIE</td>
<td></td>
<td>WRIE</td>
<td>ADRIE</td>
<td>PCIE</td>
<td>RSCIE</td>
<td>SCIE</td>
<td>588</td>
</tr>
<tr>
<td>I2CxADR0</td>
<td>ADR7</td>
<td>ADR6</td>
<td>ADR5</td>
<td>ADR4</td>
<td>ADR3</td>
<td>ADR2</td>
<td>ADR1</td>
<td>ADR0</td>
<td>589</td>
</tr>
<tr>
<td>I2CxADR1</td>
<td>ADR14</td>
<td>ADR13</td>
<td>ADR12</td>
<td>ADR11</td>
<td>ADR10</td>
<td>ADR9</td>
<td>ADR8</td>
<td></td>
<td>590</td>
</tr>
<tr>
<td>I2CxADR2</td>
<td>ADR7</td>
<td>ADR6</td>
<td>ADR5</td>
<td>ADR4</td>
<td>ADR3</td>
<td>ADR2</td>
<td>ADR1</td>
<td>ADR0</td>
<td>591</td>
</tr>
<tr>
<td>I2CxADR3</td>
<td>ADR14</td>
<td>ADR13</td>
<td>ADR12</td>
<td>ADR11</td>
<td>ADR10</td>
<td>ADR9</td>
<td>ADR8</td>
<td></td>
<td>592</td>
</tr>
<tr>
<td>I2CxADB0</td>
<td>ADB7</td>
<td>ADB6</td>
<td>ADB5</td>
<td>ADB4</td>
<td>ADB3</td>
<td>ADB2</td>
<td>ADB1</td>
<td>ADB0</td>
<td>593</td>
</tr>
<tr>
<td>I2CxADB1</td>
<td>ADB7</td>
<td>ADB6</td>
<td>ADB5</td>
<td>ADB4</td>
<td>ADB3</td>
<td>ADB2</td>
<td>ADB1</td>
<td>ADB0</td>
<td>594</td>
</tr>
</tbody>
</table>

**Legend:**  
— = unimplemented, read as ‘0’. Shaded cells are unused by the I2C module.
34.0 FIXED VOLTAGE REFERENCE (FVR)

The Fixed Voltage Reference, or FVR, is a stable voltage reference, independent of VDD, 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 VDD.

34.1 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 36.0 “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 37.0 “5-Bit Digital-to-Analog Converter (DAC) Module” and Section 38.0 “Comparator Module” for additional information.

34.2 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 34-1: VOLTAGE REFERENCE BLOCK DIAGRAM**
34.3 Register Definitions: FVR Control

REGISTER 34-1:  FVRCON: FIXED VOLTAGE REFERENCE CONTROL REGISTER

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on page</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN</td>
<td>RDY</td>
<td>TSEN</td>
<td>TSRNG</td>
<td>CDAFVR&lt;1:0&gt;</td>
<td>ADFVR&lt;1:0&gt;</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

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

- EN: Fixed Voltage Reference Enable bit
  1 = Fixed Voltage Reference is enabled
  0 = Fixed Voltage Reference is disabled

- RDY: Fixed Voltage Reference Ready Flag bit
  1 = Fixed Voltage Reference output is ready for use
  0 = Fixed Voltage Reference output is not ready or not enabled

- TSEN: Temperature Indicator Enable bit
  1 = Temperature Indicator is enabled
  0 = Temperature Indicator is disabled

- TSRNG: Temperature Indicator Range Selection bit
  1 = VOUT = 3VT (High Range)
  0 = VOUT = 2VT (Low Range)

- CDAFVR<1:0>: Comparator FVR Buffer Gain Selection bits
  11 = FVR Buffer 1 Gain is 4x, (4.096V)
  10 = FVR Buffer 1 Gain is 2x, (2.048V)
  01 = FVR Buffer 1 Gain is 1x, (1.024V)
  00 = FVR Buffer 1 is off

- ADFVR<1:0>: ADC FVR Buffer Gain Selection bit
  11 = FVR Buffer 2 Gain is 4x, (4.096V)
  10 = FVR Buffer 2 Gain is 2x, (2.048V)
  01 = FVR Buffer 2 Gain is 1x, (1.024V)
  00 = FVR Buffer 2 is off

Note 1:  FVRRDY is always ‘1’.
Note 2:  Fixed Voltage Reference output cannot exceed VDD.
Note 3:  See Section 35.0 “Temperature Indicator Module” for additional information.

TABLE 34-1:  SUMMARY OF REGISTERS ASSOCIATED WITH FIXED VOLTAGE REFERENCE

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on page</th>
</tr>
</thead>
<tbody>
<tr>
<td>FVRCON</td>
<td>EN</td>
<td>RDY</td>
<td>TSEN</td>
<td>TSRNG</td>
<td>CDAFVR&lt;1:0&gt;</td>
<td>ADFVR&lt;1:0&gt;</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Legend:  — = Unimplemented location, read as ‘0’. Shaded cells are not used with the Fixed Voltage Reference.
35.0 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°C and +125°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.

35.1 Module Operation

The temperature indicator module consists of a temperature-sensing circuit that provides a voltage to the device ADC. The analog voltage output, VMEAS, varies inversely to the device temperature. The output of the temperature indicator is referred to as VMEAS.

Figure 35-1 shows a simplified block diagram of the temperature indicator module.

FIGURE 35-1: 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 36.0 “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 34.0 “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 the next section for more details on the range settings.

35.1.1 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 VDD 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 VDD voltage is needed to operate the circuit.

The output voltage of the sensor is the highest value at -40°C and the lowest value at +125°C.

High Range: The high range is selected by setting the TSRNG bit of the FVRCON register. This range may not be suitable for battery-powered applications. The ADC reading (in counts) at 90°C for the high range setting is stored in the DIA Table (Table 5-3) as parameter TSHR2.

Low Range: This mode is useful in applications in which the VDD is too low for high-range operation. The VDD in this mode can be as low as 1.8V. VDD must, however, be at least 0.5V higher than the maximum sensor voltage depending on the expected low operating temperature. The ADC reading (in counts) at 90°C for the Low range setting is stored in the DIA Table (Table 5-3) as parameter TSLR2.

35.1.2 MINIMUM OPERATING VDD

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, VDD, must be high enough to ensure that the temperature circuit is correctly biased.

Table 35-1 shows the recommended minimum VDD vs. Range setting.

<table>
<thead>
<tr>
<th>TABLE 35-1: RECOMMENDED VDD vs. RANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min. VDD, TSRNG = 1 (High Range)</td>
</tr>
<tr>
<td>≥ 2.5</td>
</tr>
</tbody>
</table>
35.2 Temperature Calculation

This section describes the steps involved in calculating 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 44-27 for the corresponding range setting.

Equation 35-1 provides an estimate for the die temperature based on the above parameters.

\[
T_{\text{MEAS}} = 90 + \frac{(ADC_{\text{MEAS}} - ADC_{\text{DIA}}) \times FVRA2X}{(2^N - 1) \times Mv}
\]

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 10 measurements of ADCMEAS to reduce noise and improve accuracy.

35.2.1 CALIBRATION

35.2.1.1 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.

35.2.2 TEMPERATURE RESOLUTION

The resolution of the ADC reading, \( M_a \) (°C/count), depends on both the ADC resolution N and the reference voltage used for conversion, as shown in Equation 35-2. It is recommended to use the smallest VREF value, such as the ADC FVR1 Output Voltage for 2x setting (FVRA2X) value from the DIA. Refer to Table 5-3 for DIA location.

Note: Refer to Table 44-19 for FVR reference voltage accuracy.

35.3 ADC Acquisition Time

To ensure accurate temperature measurements, the user must wait a certain minimum acquisition time (parameter TS01 in Table 44-27) for the ADC value to settle, after the ADC input multiplexer is connected to the temperature indicator output, before the conversion is performed.

### Table 35-2: SUMMARY OF REGISTERS ASSOCIATED WITH THE TEMPERATURE INDICATOR

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on page</th>
</tr>
</thead>
<tbody>
<tr>
<td>FVRCON</td>
<td>EN</td>
<td>RDY</td>
<td>TSEN</td>
<td>TSRNG</td>
<td>CDAFVR&lt;1:0&gt;</td>
<td>ADFVR&lt;1:0&gt;</td>
<td></td>
<td></td>
<td>597</td>
</tr>
</tbody>
</table>

Legend: — = Unimplemented location, read as ‘0’. Shaded cells are unused by the temperature indicator module.

Note 1: It is recommended to take the average of ten measurements of ADCMEAS to reduce noise and improve accuracy.
36.0 ANALOG-TO-DIGITAL CONVERTER WITH COMPUTATION (ADC²) MODULE

The Analog-to-Digital Converter with Computation (ADC²) 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 36-1 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 36-1: ADC² BLOCK DIAGRAM

- **Vref⁺, Vref⁻**: Reference voltages
- **Enable**: Enables the ADC
- **DACx_output**: Output of DAC
- **FVR_buffer**: Voltage reference buffer
- **Temp Indicator**: Displays temperature indicator
- **PCH<5:0>**: Prescaler channels
- **External Channel Inputs**: AN0, ANa, ANz
- **Internal Channel Inputs**: Fosc, Divider
- **GO/DONE**: Start/Stop of ADC conversion
- **ADON**: Auto conversion enable
- **Trigger Select**: Source selection
- **12-bit Result**: ADC output
- **ADRESH, ADRESL**: 16-bit result
- **ADC CLOCK SOURCE**: Selects clock source
- **FOSC, FRC**: Oscillator modes
- **CS**: Chip select signal

**Legend:***
- **VSS**: Ground
- **VDD**: Power supply
- **VREF⁺**: Reference input
- **PREF<1:0>**: Reference configuration
- **ADIF**: ADC interrupt flag
- **ADFM**: ADC full scale flag

**Additional Notes:**
- Positive reference select
- Selects between VREF⁺ and VREF⁻

**Revision:**
- Rev. 1.000 034D
- 11/2/2016
36.1 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

36.1.1 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.

36.1.2 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>)
- Eight PORTD pins (RD<7:0>, PIC18(L)F45/46/47/55/56/57K42 only)
- Three PORTE pins (RE<2:0>, PIC18(L)F45/46/47/55/56/57K42 only)
- Eight PORTF pins (RF<7:0>, PIC18(L)F55/56/57K42 only)
- 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 36.2 “ADC Operation” for more information.

36.1.3 ADC VOLTAGE REFERENCE

The PREF<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 NREF bit of the ADREF register provides control of the negative voltage reference. The negative voltage reference can be:

- VREF- pin
- Vss

See Section 34.0 “Fixed Voltage Reference (FVR)” for more details on the Fixed Voltage Reference.

36.1.4 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^n)(where n is from 1 to 128)
- FRC (dedicated RC oscillator)

The time to complete one bit conversion is defined as TAD. Refer to Figure 36-2 for the complete timing details of the ADC conversion.

For correct conversion, the appropriate TAD specification must be met. Refer to Table 44-16 for more information. Table 36-1 gives examples of appropriate ADC clock selections.

![Note: Analog voltages on any pin that is defined as a digital input may cause the input buffer to conduct excess current.]

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 36-1: ADC CLOCK PERIOD (TA0) VS. DEVICE OPERATING FREQUENCIES(1,4)

<table>
<thead>
<tr>
<th>ADC Clock Source</th>
<th>CS&lt;5:0&gt;</th>
<th>64 MHz</th>
<th>32 MHz</th>
<th>20 MHz</th>
<th>16 MHz</th>
<th>8 MHz</th>
<th>4 MHz</th>
<th>1 MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fosc/2</td>
<td>000000</td>
<td>31.25 ns(2)</td>
<td>62.5 ns(2)</td>
<td>100 ns(2)</td>
<td>125 ns(2)</td>
<td>250 ns(2)</td>
<td>500 ns(2)</td>
<td>2.0 µs</td>
</tr>
<tr>
<td>Fosc/4</td>
<td>000001</td>
<td>62.5 ns(2)</td>
<td>125 ns(2)</td>
<td>200 ns(2)</td>
<td>250 ns(2)</td>
<td>500 ns(2)</td>
<td>1.0 µs</td>
<td>4.0 µs</td>
</tr>
<tr>
<td>Fosc/6</td>
<td>000010</td>
<td>125 ns(2)</td>
<td>187.5 ns(2)</td>
<td>300 ns(2)</td>
<td>375 ns(2)</td>
<td>750 ns(2)</td>
<td>1.5 µs</td>
<td>6.0 µs</td>
</tr>
<tr>
<td>Fosc/8</td>
<td>000011</td>
<td>187.5 ns(2)</td>
<td>250 ns(2)</td>
<td>400 ns(2)</td>
<td>500 ns(2)</td>
<td>1.0 µs</td>
<td>2.0 µs</td>
<td>8.0 µs</td>
</tr>
</tbody>
</table>
| ...              | ...     | ...     | ...     | ...     | ...     | ...    | ...    | ...
| Fosc/16          | 000111  | 250 ns(2) | 500 ns(2) | 800 ns(2) | 1.0 µs | 2.0 µs | 4.0 µs | 16.0 µs(3) |
| ...              | ...     | ...     | ...     | ...     | ...     | ...    | ...    | ...
| Fosc/128         | 111111  | 2.0 µs | 4.0 µs | 6.4 µs | 8.0 µs | 16.0 µs(3) | 32.0 µs(2) | 128.0 µs(2) |
| 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 |

**Legend:**  Shaded cells are outside of recommended range.

**Note 1:** See TA0 parameter for FRC source typical TA0 value.

**Note 2:** These values violate the required TA0 time.

**Note 3:** Outside the recommended TA0 time.

**Note 4:** The ADC clock period (TA0) 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 36-2: ANALOG-TO-DIGITAL CONVERSION CYCLES

- **Precharge Time:** 1-8191 FOSC (TPRE)
- **Acquisition/Sharing Time:** 1-8191 FOSC (TACQ)
- **ADC Conversion Time**

- **External and Internal Channels are charged/discharged**
- **External and Internal Channels share charge**
- **Holding capacitor CHOLD is disconnected from analog input**

On the following cycle:

- ADRES is loaded
- GO bit is cleared
- ADIF bit is set

**Set GO bit**

**Note 1:** Refer to Table 44-16 for ADC Conversion Timing Specifications.
36.1.5 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 PIRx register. The ADC Interrupt Enable is the ADIE bit in the PIEx 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 PIEx 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.

36.1.6 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 36-3 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 36-3: 12-BIT ADC CONVERSION RESULT FORMAT**
36.2 ADC Operation

36.2.1 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 36-3)
- A continuous-mode retrigger (see section Section 36.6.8 “Continuous Sampling mode”)

Note: The GO bit should not be set in the same instruction that turns on the ADC. Refer to Section 36.2.6 “ADC Conversion Procedure (Basic Mode)”.

36.2.2 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 MATH 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.

36.2.3 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.

36.2.4 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.

Note: The GO bit should not be set in the same instruction that turns on the ADC. Refer to Section 36.2.6 “ADC Conversion Procedure (Basic Mode)”.
36.2.5 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 36-33 for auto-conversion sources.

36.2.6 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\(^1\)

4. If ADACQ = 0, software must wait the required acquisition time\(^2\).

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).

**EXAMPLE 36-1: ADC CONVERSION**

```c
/*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
    }
}
```

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 36.3 “ADC Acquisition Requirements”.

36.3 ADC Acquisition Requirements

For the ADC to meet its specified accuracy, the charge holding capacitor (C\text{HOLD}) must be allowed to fully charge to the input channel voltage level. The Analog Input model is shown in Figure 36-4. The source impedance (R\text{S}) and the internal sampling switch (RSS) impedance directly affect the time required to charge the capacitor C\text{HOLD}. The sampling switch (RSS) impedance varies over the device voltage (V\text{DD}), refer to Figure 36-4. Refer to Parameter AD08 mentioned in Table 44-15 for the maximum recommended impedance for analog sources. 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 36-1 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 36-1: ACQUISITION TIME EXAMPLE**

Assumptions: Temperature = 50°C and external impedance of 1kΩ 5.0V V\text{DD}

\[ T_{ACQ} = \text{Amplifier Settling Time} + \text{Hold Capacitor Charging Time} + \text{Temperature Coefficient} \]
\[ = T_{AMP} + T_{C} + T_{COFF} \]
\[ = 2\mu s + T_{C} + [(\text{Temperature} - 25^\circ C)(0.05\mu s/^\circ C)] \]

The value for \( T_{C} \) can be approximated with the following equations:

\[ V_{APPLIED}\left(1 - \frac{1}{2^{n+1}}\right) = V_{CHOLD} \quad \text{;[1] \ V_{CHOLD} charged to within 1/2 \ lsb} \]

\[ V_{APPLIED}\left(1 - e^{-\frac{T_{C}}{R_{C}}}\right) = V_{CHOLD} \quad \text{;[2] \ V_{CHOLD} charge response to V_{APPLIED}} \]

\[ V_{APPLIED}\left(1 - e^{-\frac{T_{C}}{R_{C}}}\right) = V_{APPLIED}\left(1 - \frac{1}{2^{n+1}}\right) \quad \text{;combining [1] and [2]} \]

*Note: Where \( n \) = number of bits of the ADC.*

Solving for \( T_{C} \):

\[ T_{C} = -CHOLD(R_{IC} + RSS + R_{S}) \ln(1/8191) \]
\[ = -28pF(1k\Omega + 7k\Omega + 1k\Omega) \ln(0.0001221) \]
\[ = 2.27\mu s \]

Therefore:

\[ T_{ACQ} = 2\mu s + 2.27\mu s + [(50^\circ C - 25^\circ C)(0.05\mu s/^\circ C)] \]
\[ = 5.52\mu s \]

**Note 1:** The reference voltage (V\text{REF}) has no effect on the equation, since it cancels itself out.

**2:** The charge holding capacitor (C\text{HOLD}) is not discharged after each conversion.

**3:** The maximum recommended impedance for analog sources is mentioned in Parameter AD08 in Table 44-15. This is required to meet the pin leakage specification.
FIGURE 36-4: ANALOG INPUT MODEL

Legend:
- **CHOLD** = Sample/Hold Capacitance
- **CPIN** = Input Capacitance
- **I Leakage** = Leakage current at the pin due to various junctions
- **RIC** = Interconnect Resistance
- **RSS** = Resistance of Sampling Switch
- **SS** = Sampling Switch
- **VT** = Threshold Voltage

**Note 1:** Refer to Table 44-6 (parameter D340 and D341).

FIGURE 36-5: ADC TRANSFER FUNCTION

Legend:
- **REF-** = Full-Scale Range
- **REF+** = 0.5 LSB
- **00h** = Zero-Scale Transition
- **01h** = Full-Scale Transition
- **02h** = 1.5 LSB
- **03h** = Analog Input Voltage

Legend:
- **ADC Output Code**
- **Full-Scale Range**
- **0.5 LSB**
- **1.5 LSB**
- **REF-**
- **REF+**
36.4 ADC Charge Pump

The ADC module has a dedicated charge pump which can be controlled through the ADCP register (Register 36-36). 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.

36.5 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 36-6 shows the basic block diagram of the CVD portion of the ADC module.

**FIGURE 36-6: HARDWARE CAPACITIVE VOLTAGE DIVIDER BLOCK DIAGRAM**
36.5.1 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 re-connected, 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 36-7 shows the waveform for two inverted CVD measurements, which is known as differential CVD measurement.

FIGURE 36-7: DIFFERENTIAL CVD MEASUREMENT WAVEFORM
36.5.2 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 ADPRE 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 ADPRE 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 PPOL 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 PPOL bit of ADCON1. The amount of time that this charging receives is controlled by the ADPRE register.

36.5.3 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 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.

36.5.4 GUARD RING OUTPUTS

Figure 36-8 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 36-8 and Figure 36-9.

FIGURE 36-8: GUARD RING CIRCUIT
36.5.5 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 36-10.
36.6 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 36-10: COMPUTATIONAL FEATURES SIMPLIFIED BLOCK DIAGRAM

The operation of the ADC computational features is controlled by ADMD <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 (ADDSEN = 0) or double (ADDSEN = 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 36-2 below.
### TABLE 36-2: COMPUTATION MODES

<table>
<thead>
<tr>
<th>Mode</th>
<th>ADMD</th>
<th>Bit Clear Conditions</th>
<th>Value after Trigger completion</th>
<th>Threshold Operations</th>
<th>Value at ADTIF interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ACC and CNT</td>
<td>ACC</td>
<td>CNT</td>
<td>Retrigger</td>
</tr>
<tr>
<td>Basic</td>
<td>0</td>
<td>ADACLR = 1</td>
<td>Unchanged</td>
<td>Unchanged</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accumulate</td>
<td>1</td>
<td>ADACLR = 1</td>
<td>S + ACC or (S2-S1) + ACC</td>
<td>If (CNT=0xFF): CNT, otherwise: CNT+1</td>
<td>No</td>
</tr>
<tr>
<td>Average</td>
<td>2</td>
<td>ADACLR = 1 or CNT&gt;=RPT at GO or retrigger</td>
<td>S + ACC or (S2-S1) + ACC</td>
<td>If (CNT=0xFF): CNT, otherwise: CNT+1</td>
<td>No</td>
</tr>
<tr>
<td>Burst Average</td>
<td>3</td>
<td>ADACLR = 1 or GO set or retrigger</td>
<td>Each repetition: same as Average End with sum of all samples</td>
<td>Each repetition: same as Average End with CNT=RPT</td>
<td>Repeat while CNT&lt;RPT</td>
</tr>
<tr>
<td>Low-pass Filter</td>
<td>4</td>
<td>ADACLR = 1</td>
<td>S+ACC-ACC/2ADCRS or (S2-S1)+ACC-ACC/2ADCRS</td>
<td>Count up, stop counting when CNT = 0xFF</td>
<td>No</td>
</tr>
</tbody>
</table>

**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.
36.6.1 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 \(2^{\text{accumulator_width}}-1 = 2^{16} = 262143\), 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 36-3 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>
<thead>
<tr>
<th>ADCRS</th>
<th>(\omega_T) (radians) @ -3 dB Frequency</th>
<th>dB @ F(_{\text{nyquist}}=1/(2T))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.72</td>
<td>-9.5</td>
</tr>
<tr>
<td>2</td>
<td>0.284</td>
<td>-16.9</td>
</tr>
<tr>
<td>3</td>
<td>0.134</td>
<td>-23.5</td>
</tr>
<tr>
<td>4</td>
<td>0.065</td>
<td>-29.8</td>
</tr>
<tr>
<td>5</td>
<td>0.032</td>
<td>-36.0</td>
</tr>
<tr>
<td>6</td>
<td>0.016</td>
<td>-42.0</td>
</tr>
<tr>
<td>7</td>
<td>0.0078</td>
<td>-48.1</td>
</tr>
</tbody>
</table>

36.6.2 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.

36.6.3 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 into 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 36.6.7 “Threshold Comparison”) and the ADTIF interrupt may trigger.

36.6.4 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\(^C\)NT, 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.
36.6.5 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 36.6.8 “Continuous Sampling mode”) is not enabled. This allows for a threshold comparison on the average of a short burst of ADC samples.

36.6.6 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 36-2 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 36-3).

36.6.7 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 36-4 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 or equal to 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.

36.6.8 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 re-triggering occurs automatically.

If ADSOI = 1, a threshold interrupt condition will clear GO and the conversions will stop.

36.6.9 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).
### Register Definitions: ADC Control

#### REGISTER 36-1: ADCON0: ADC CONTROL REGISTER 0

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>State</th>
<th>Notes</th>
</tr>
</thead>
</table>
| 7   | ON          | ADC Enable bit | 1 = ADC is enabled, 0 = ADC is disabled. 
|     | 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. |
| 5   | Unimplemented | Read as '0' |
| 4   | CS          | ADC Clock Selection bit | 1 = Clock supplied from FRC dedicated oscillator, 0 = Clock supplied by Fosc, divided according to ADCLK register. |
| 3   | Unimplemented | Read as '0' |
| 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. |
| 1   | Unimplemented | Read as '0' |
| 0   | GO          | ADC Conversion Status bit(1) | 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. |

**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

**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 36-2: ADCON1: ADC CONTROL REGISTER 1

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>DSEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPOL</td>
<td>IPEN</td>
<td>GPOL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
</tbody>
</table>

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**:

<table>
<thead>
<tr>
<th><strong>PPOL</strong></th>
<th>Action During 1st Precharge Stage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><strong>External</strong> (selected analog I/O pin) Connected to Vdd</td>
</tr>
<tr>
<td></td>
<td><strong>Internal</strong> (AD sampling capacitor) $C_{HOLD}$ connected to Vss</td>
</tr>
<tr>
<td>0</td>
<td><strong>External</strong> (selected analog I/O pin) Connected to Vss</td>
</tr>
<tr>
<td></td>
<td><strong>Internal</strong> (AD sampling capacitor) $C_{HOLD}$ connected to Vdd</td>
</tr>
</tbody>
</table>

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 36-3: ADCON2: ADC CONTROL REGISTER 2

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>PSIS</td>
<td>ADC Previous Sample Input Select bits</td>
</tr>
<tr>
<td>6-4</td>
<td>CRS&lt;2:0&gt;</td>
<td>ADC Accumulated Calculation Right Shift Select bits</td>
</tr>
<tr>
<td>3</td>
<td>ACLR</td>
<td>A/D Accumulator Clear Command bit</td>
</tr>
<tr>
<td>2-0</td>
<td>MD&lt;2:0&gt;</td>
<td>ADC Operating Mode Selection bits</td>
</tr>
</tbody>
</table>

### Legend:
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as '0'
- **u** = Bit is unchanged
- **x** = Bit is unknown
- **'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 $2^{ADC_RS}$, filter gain is 1:1
- If ADMD = **001**, **010**, or **011**:
  - The accumulated value is right-shifted by CRS (divided by $2^{ADC_RS}$) $^{(1,2)}$
- Otherwise:
  - Bits are ignored

#### bit 3
**ACLR:** A/D Accumulator Clear Command bit $^{(3)}$
- **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 $^{(4)}$
- **111** = 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 $2^{ADC_RS}$.
2. $^{(1)}$ $ADCRS = 3'b111$ 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 36-2 for Full mode descriptions.
## REGISTER 36-4: ADCON3: ADC CONTROL REGISTER 3

<table>
<thead>
<tr>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>CALC&lt;2:0&gt;</td>
<td>SOI</td>
<td>TMD&lt;2:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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**

<table>
<thead>
<tr>
<th>CALC&lt;2:0&gt;</th>
<th>DSEN = 0 Single-Sample Mode</th>
<th>DSEN = 1 CVD Double-Sample Mode&lt;sup&gt;(1)&lt;/sup&gt;</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>110</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>101</td>
<td>FLTR-STPT</td>
<td>FLTR-STPT</td>
<td>Average/filtered value vs. setpoint</td>
</tr>
<tr>
<td>100</td>
<td>PREV-FLTR</td>
<td>PREV-FLTR</td>
<td>First derivative of filtered value&lt;sup&gt;(3)&lt;/sup&gt; (negative)</td>
</tr>
<tr>
<td>011</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>010</td>
<td>RES-FLTR (RES-PREV)-FLTR</td>
<td>(RES-PREV)-FLTR</td>
<td>Actual result vs. averaged/filtered value</td>
</tr>
<tr>
<td>001</td>
<td>RES-STPT (RES-PREV)-STPT</td>
<td>(RES-PREV)-STPT</td>
<td>Actual result vs.setpoint</td>
</tr>
<tr>
<td>000</td>
<td>RES-PREV</td>
<td>RES-PREV</td>
<td>First derivative of single measurement&lt;sup&gt;(2)&lt;/sup&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Actual CVD result in CVD mode&lt;sup&gt;(2)&lt;/sup&gt;</td>
</tr>
</tbody>
</table>

**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 retriggered
- **0** = GO is not cleared by hardware, must be cleared by software to stop retrigger

**bit 2-0**

**TMD<2:0>: Threshold Interrupt Mode Select bits**

<table>
<thead>
<tr>
<th>TMD&lt;2:0&gt;</th>
<th>ASCII</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>111</td>
<td>Interrupt regardless of threshold test results</td>
</tr>
<tr>
<td>110</td>
<td>110</td>
<td>Interrupt if ERR&gt;UTH</td>
</tr>
<tr>
<td>101</td>
<td>101</td>
<td>Interrupt if ERR&lt;UTH</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>Interrupt if ERR&lt;LTH or ERR&gt;UTH</td>
</tr>
<tr>
<td>011</td>
<td>011</td>
<td>Interrupt if ERR&gt;LTH and ERR&lt;UTH</td>
</tr>
<tr>
<td>010</td>
<td>010</td>
<td>Interrupt if ERR&gt;LTH</td>
</tr>
<tr>
<td>001</td>
<td>001</td>
<td>Interrupt if ERR&lt;LTH</td>
</tr>
<tr>
<td>000</td>
<td>000</td>
<td>Never interrupt</td>
</tr>
</tbody>
</table>

**Note 1:** When PSIS = 0, the value of (RES-PREV) is the value of (S2-S1) from Table 36-2.

**2:** When PSIS = 0

**3:** When PSIS = 1.
### REGISTER 36-5: ADSTAT: ADC STATUS REGISTER

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>AOV: ADC Accumulator Overflow bit</td>
<td>1 = ADC accumulator or ADC filter or ERR calculation have overflowed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 = ADC accumulator, ADC filter and ERR calculation have not overflowed</td>
</tr>
<tr>
<td>6</td>
<td>UTHR: ADC Module Greater-than Upper Threshold Flag bit</td>
<td>1 = ERR &gt; UTH</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 = ERR ≤ UTH</td>
</tr>
<tr>
<td>5</td>
<td>LTHR: ADC Module Less-than Lower Threshold Flag bit</td>
<td>1 = ERR &lt; LTH</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 = ERR ≥ LTH</td>
</tr>
<tr>
<td>4</td>
<td>MATH: ADC Module Computation Status bit</td>
<td>1 = Registers ACC, FLTR, UTH, LTH and the AOV bit are updating or have already updated</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 = Associated registers/bits have not changed since this bit was last cleared</td>
</tr>
<tr>
<td>3</td>
<td>Unimplemented: Read as ‘0’</td>
<td></td>
</tr>
<tr>
<td>2-0</td>
<td>STAT&lt;2:0&gt;: ADC Module Cycle Multistage Status bits</td>
<td>111 = ADC module is in 2(^{nd}) conversion stage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>110 = ADC module is in 2(^{nd}) acquisition stage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>101 = ADC module is in 2(^{nd}) precharge stage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100 = Not used</td>
</tr>
<tr>
<td></td>
<td></td>
<td>011 = ADC module is in 1(^{st}) conversion stage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>010 = ADC module is in 1(^{st}) acquisition stage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>001 = ADC module is in 1(^{st}) precharge stage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>000 = ADC module is not converting</td>
</tr>
</tbody>
</table>

**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

**Note 1:** If CS = 1, and FOSC < FRC, these bits may be invalid.
### REGISTER 36-6: ADCLK: ADC CLOCK SELECTION REGISTER

<table>
<thead>
<tr>
<th></th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>U-0</td>
<td>U-0</td>
<td>—</td>
<td>—</td>
<td>CS&lt;5:0&gt;</td>
<td>bit 7</td>
<td>bit 0</td>
<td></td>
</tr>
</tbody>
</table>

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

- **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 36-7: ADREF: ADC REFERENCE SELECTION REGISTER

<table>
<thead>
<tr>
<th></th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>NREF</td>
<td>PREF&lt;1:0&gt;</td>
</tr>
</tbody>
</table>

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

- **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 36-8: ADPCH: ADC POSITIVE CHANNEL SELECTION REGISTER**

<table>
<thead>
<tr>
<th>bit 7-6</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Unimplemented:</td>
<td>Read as ‘0’</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 5-0</td>
<td>ADPCH&lt;5:0&gt;: ADC Positive Input Channel Selection bits</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| 111111 = FVR buffer 2(2) | 010111 = ANC7 |
| 111110 = FVR buffer 1(2) | 010110 = ANC6 |
| 111101 = DAC1 output(1) | 010101 = ANC5 |
| 111100 = Temperature indicator(3) | 010100 = ANC4 |
| 111011 = Vss (Analog Ground) | 010011 = ANC3 |
| 111010 = Reserved. No channel connected. | 010010 = ANC2 |
| • | 010001 = ANC1 |
| • | 010000 = ANC0 |
| • | 001111 = ANB7 |
| 110000 = Reserved. No channel connected. | 001110 = ANB6 |
| 101111 = ANF7(4) | 001101 = ANB5 |
| 101110 = ANF6(4) | 001100 = ANB4 |
| 101101 = ANF5(4) | 001111 = ANA7 |
| 101100 = ANF4(4) | 001110 = ANA6 |
| 101011 = ANF3(4) | 001011 = ANA5 |
| 101010 = ANF2(4) | 001010 = ANA4 |
| 101001 = ANF1(4) | 001001 = ANA3 |
| 101000 = ANF0(4) | 001000 = ANA2 |
| 100111 = Reserved. No channel connected. | 000011 = ANA1 |
| • | 000010 = ANA0 |
| • | 000001 = ANA1 |
| 100011 = Reserved. No channel connected. | 000000 = ANA0 |
| 100010 = ANE2(5) | 011111 = AND7(5) |
| 100001 = ANE1(5) | 011110 = AND6(5) |
| 100000 = ANE0(5) | 011101 = AND5(5) |
| 011111 = AND7(5) | 011100 = AND4(5) |
| 011101 = AND6(5) | 011011 = AND3(5) |
| 011100 = AND5(5) | 011010 = AND2(5) |
| 011011 = AND3(5) | 011001 = AND1(5) |
| 011010 = AND2(5) | 011000 = AND0(5) |

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

**Note:**
1. See Section 37.0 “5-Bit Digital-to-Analog Converter (DAC) Module” for more information.
2. See Section 34.0 “Fixed Voltage Reference (FVR)” for more information.
3. See Section 35.0 “Temperature Indicator Module” for more information.
5. Reserved on PIC18(L)F26/27K42 parts.
REGISTER 36-9: ADPREL: ADC PRECHARGE TIME CONTROL REGISTER (LOW BYTE)

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PRE&lt;7:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

bit 7-0  PRE<7:0>: Precharge Time Select bits
See Table 36-4.

REGISTER 36-10: ADPRE: ADC PRECHARGE TIME CONTROL REGISTER (HIGH BYTE)

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>PRE&lt;12:8&gt;</td>
</tr>
</tbody>
</table>

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

bit 7-5  Unimplemented: Read as ‘0’
bit 4-0  PRE<12:8>: Precharge Time Select bits
See Table 36-4.

Note: If PRE is not equal to ‘0’, then ADACQ = b’00000000 means Acquisition time is 256 clocks of the selected ADC clock.

TABLE 36-4: PRECHARGE TIME

<table>
<thead>
<tr>
<th>ADPRE</th>
<th>Precharge time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1111 1111 1111</td>
<td>8191 clocks of the selected ADC clock</td>
</tr>
<tr>
<td>1 1111 1111 1110</td>
<td>8190 clocks of the selected ADC clock</td>
</tr>
<tr>
<td>1 1111 1111 1101</td>
<td>8189 clocks of the selected ADC clock</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>0 0000 0000 0010</td>
<td>2 clocks of the selected ADC clock</td>
</tr>
<tr>
<td>0 0000 0000 0001</td>
<td>1 clock of the selected ADC clock</td>
</tr>
<tr>
<td>0 0000 0000 0000</td>
<td>Not included in the data conversion cycle</td>
</tr>
</tbody>
</table>
REGISTER 36-11: ADACQL: ADC ACQUISITION TIME CONTROL REGISTER (LOW BYTE)

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACQ&lt;7:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as ‘0’
- u = Bit is unchanged
- x = Bit is unknown
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared

bit 7-0  ACQ<7:0>: Acquisition (charge share time) Select bits
See Table 36-5.

REGISTER 36-12: ADACQH: ADC ACQUISITION TIME CONTROL REGISTER (HIGH BYTE)

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACQ&lt;12:8&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as ‘0’
- u = Bit is unchanged
- x = Bit is unknown
- ‘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 36-5.

TABLE 36-5: ACQUISITION TIME

<table>
<thead>
<tr>
<th>ADACQ</th>
<th>ACQUISITION TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1111 1111 1111</td>
<td>8191 clocks of the selected ADC clock</td>
</tr>
<tr>
<td>1 1111 1111 1110</td>
<td>8190 clocks of the selected ADC clock</td>
</tr>
<tr>
<td>1 1111 1111 1101</td>
<td>8189 clocks of the selected ADC clock</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>0 0000 0000 0010</td>
<td>2 clocks of the selected ADC clock</td>
</tr>
<tr>
<td>0 0000 0000 0001</td>
<td>1 clock of the selected ADC clock</td>
</tr>
<tr>
<td>0 0000 0000 0000</td>
<td>Not included in the data conversion cycle(1)</td>
</tr>
</tbody>
</table>

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 36-13: ADCAP: ADC ADDITIONAL SAMPLE CAPACITOR SELECTION REGISTER

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 7-5</td>
<td>Unimplemented: Read as ‘0’</td>
<td>bit 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 ADCAP<4:0>: ADC Additional Sample Capacitor Selection bits
11111 = 31 pF
11110 = 30 pF
11101 = 29 pF
00011 = 3 pF
00010 = 2 pF
00001 = 1 pF
00000 = No additional capacitance

REGISTER 36-14: ADRPT: ADC REPEAT SETTING REGISTER

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPT&lt;7:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

bit 7-0 RPT<7:0>: ADC Repeat Threshold 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 36-2 for more details.
REGISTER 36-15: ADCNT: ADC REPEAT COUNTER REGISTER

<table>
<thead>
<tr>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CNT&lt;7:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 7</td>
<td>bit 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:
R = Readable bit
W = Writable bit
U = Unimplemented bit, read as ‘0’
u = Bit is unchanged
x = Bit is unknown
-\text{n/n} = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set
‘0’ = Bit is cleared

bit 7-0  \text{CNT<7:0>: ADC Repeat Count bits}
Counts the number of times that the ADC has been triggered and is used along with CNT to determine when the error threshold is checked when the computation is Low-pass Filter, Burst Average, or Average modes. See Table Table 36-2 for more details.

REGISTER 36-16: ADFLTRH: ADC FILTER HIGH BYTE REGISTER

<table>
<thead>
<tr>
<th>R-x</th>
<th>R-x</th>
<th>R-x</th>
<th>R-x</th>
<th>R-x</th>
<th>R-x</th>
<th>R-x</th>
<th>R-x</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLTR&lt;15:8&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 7</td>
<td>bit 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:
R = Readable bit
W = Writable bit
U = Unimplemented bit, read as ‘0’
u = Bit is unchanged
x = Bit is unknown
-\text{n/n} = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set
‘0’ = Bit is cleared

bit 7-0  \text{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 36-17: ADFLTRL: ADC FILTER LOW BYTE REGISTER

<table>
<thead>
<tr>
<th>R-x</th>
<th>R-x</th>
<th>R-x</th>
<th>R-x</th>
<th>R-x</th>
<th>R-x</th>
<th>R-x</th>
<th>R-x</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLTR&lt;7:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 7</td>
<td>bit 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:
R = Readable bit
W = Writable bit
U = Unimplemented bit, read as ‘0’
u = Bit is unchanged
x = Bit is unknown
-\text{n/n} = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set
‘0’ = Bit is cleared

bit 7-0  \text{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 36-18: ADRESH: ADC RESULT REGISTER HIGH, FM = 0

<table>
<thead>
<tr>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>ADRES&lt;11:4&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td>bit 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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

**bit 7-0**  
**ADRESH<11:4>: ADC Result Register bits**  
Upper eight bits of 12-bit conversion result.

### REGISTER 36-19: ADRESL: ADC RESULT REGISTER LOW, FM = 0

<table>
<thead>
<tr>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>ADRES&lt;3:0&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

**bit 7-4**  
**ADRESL<3:0>: ADC Result Register bits. Lower four bits of 12-bit conversion result.**

**bit 3-0**  
**Reserved**
REGISTER 36-20: ADRESH: ADC RESULT REGISTER HIGH, FM = 1

<table>
<thead>
<tr>
<th>bit 7-4</th>
<th>bit 3-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>ADRES&lt;11:8&gt;: ADC Sample Result bits. Upper four bits of 12-bit conversion result.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

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

REGISTER 36-21: ADRESL: ADC RESULT REGISTER LOW, FM = 1

<table>
<thead>
<tr>
<th>bit 7-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADRES&lt;7:0&gt;: ADC Result Register bits. Lower eight bits of 12-bit conversion result.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td>bit 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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
**REGISTER 36-22: ADPREVH: ADC PREVIOUS RESULT REGISTER**

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 6</th>
<th>bit 5</th>
<th>bit 4</th>
<th>bit 3</th>
<th>bit 2</th>
<th>bit 1</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV&lt;15:8&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

- **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

**Note 1:** If **ADPSIS = 0,** ADPREVH and ADPREVL are formatted the same way as ADRES is, depending on the FM bit.

**REGISTER 36-23: ADPREVL: ADC PREVIOUS RESULT REGISTER**

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 6</th>
<th>bit 5</th>
<th>bit 4</th>
<th>bit 3</th>
<th>bit 2</th>
<th>bit 1</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV&lt;7:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

- **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

**Note 1:** If **ADPSIS = 0,** ADPREVH and ADPREVL are formatted the same way as ADRES is, depending on the FM bit.
**REGISTER 36-24: ADACCU: ADC ACCUMULATOR REGISTER UPPER**

<table>
<thead>
<tr>
<th></th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ACC&lt;17:16&gt;</td>
</tr>
</tbody>
</table>

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

bit 7-2 **Unimplemented**: Read as ‘0’

bit 1-0 **ACC<17:16>**: ADC Accumulator MSB. Upper two bits of accumulator value. See Table 36-2 for more details.

**REGISTER 36-25: ADACCH: ADC ACCUMULATOR REGISTER HIGH**

<table>
<thead>
<tr>
<th></th>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td>ACC&lt;15:8&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

bit 7-0 **ACC<15:8>**: ADC Accumulator middle bits. Middle eight bits of accumulator value. See Table 36-2 for more details.

**REGISTER 36-26: ADACCL: ADC ACCUMULATOR REGISTER LOW**

<table>
<thead>
<tr>
<th></th>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td>ACC&lt;7:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

bit 7-0 **ACC<7:0>**: ADC Accumulator LSB. Lower eight bits of accumulator value. See Table 36-2 for more details.
**REGISTER 36-27: ADSTPTH: ADC THRESHOLD SETPOINT REGISTER HIGH**

<table>
<thead>
<tr>
<th>bit</th>
<th>STPT&lt;15:8&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

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

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 36-29 for more details.

**REGISTER 36-28: ADSTPTL: ADC THRESHOLD SETPOINT REGISTER LOW**

<table>
<thead>
<tr>
<th>bit</th>
<th>STPT&lt;7:0&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

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

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 36-30 for more details.
REGISTER 36-29: ADERRH: ADC SETPOINT ERROR REGISTER HIGH

<table>
<thead>
<tr>
<th>R-x</th>
<th>R-x</th>
<th>R-x</th>
<th>R-x</th>
<th>R-x</th>
<th>R-x</th>
<th>R-x</th>
<th>R-x</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 7</td>
<td>bit 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as ‘0’
u = Bit is unchanged
x = Bit is unknown
‘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 36-4 for more details.

REGISTER 36-30: ADERRL: ADC SETPOINT ERROR LOW BYTE REGISTER

<table>
<thead>
<tr>
<th>R-x</th>
<th>R-x</th>
<th>R-x</th>
<th>R-x</th>
<th>R-x</th>
<th>R-x</th>
<th>R-x</th>
<th>R-x</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 7</td>
<td>bit 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as ‘0’
u = Bit is unchanged
x = Bit is unknown
‘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 36-4 for more details.

REGISTER 36-31: ADLTHH: ADC LOWER THRESHOLD HIGH BYTE REGISTER

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 7</td>
<td>bit 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as ‘0’
u = Bit is unchanged
x = Bit is unknown
‘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 36-32: ADLTHL: ADC LOWER THRESHOLD LOW BYTE REGISTER

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>LTH&lt;7:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 7-0  
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 36-33: ADUTHH: ADC UPPER THRESHOLD HIGH BYTE REGISTER

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTH&lt;15:8&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 7-0  
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 36-34: ADUTHL: ADC UPPER THRESHOLD LOW BYTE REGISTER

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTH&lt;7:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 7-0  
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 36-35: ADACT: ADC AUTO CONVERSION TRIGGER CONTROL REGISTER

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

- **ACT<4:0>**

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

### 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 36-36: ADCP: ADC CHARGE PUMP CONTROL REGISTER

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPON</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CPRDY</td>
</tr>
</tbody>
</table>

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)

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

TABLE 36-6: SUMMARY OF REGISTERS ASSOCIATED WITH ADC

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADCON0</td>
<td>ON</td>
<td>CONT</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>GO</td>
<td>617</td>
</tr>
<tr>
<td>ADCON1</td>
<td>ADPPOL</td>
<td>ADIPEN</td>
<td>ADGPN</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>618</td>
</tr>
<tr>
<td>ADCON2</td>
<td>ADPSIS</td>
<td>ADCRS&lt;2:0&gt;</td>
<td>—</td>
<td>ADACLR</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ADDSEN</td>
<td>619</td>
</tr>
<tr>
<td>ADCON3</td>
<td>—</td>
<td>ADCALC&lt;2:0&gt;</td>
<td>—</td>
<td>ADSOI</td>
<td>—</td>
<td>—</td>
<td>ADTMD&lt;2:0&gt;</td>
<td>620</td>
<td></td>
</tr>
<tr>
<td>ADSTAT</td>
<td>ADAOV</td>
<td>ADUTHR</td>
<td>ADLTHR</td>
<td>ADMI</td>
<td>ADCALC&lt;3:0&gt;</td>
<td>—</td>
<td>—</td>
<td>621</td>
<td></td>
</tr>
<tr>
<td>ADCLK</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CS&lt;5:0&gt;</td>
<td>622</td>
</tr>
<tr>
<td>ADREF</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ADNREF</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ADPREF&lt;1:0&gt;</td>
</tr>
<tr>
<td>ADPCH</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ADPCH&lt;5:0&gt;</td>
<td>623</td>
</tr>
<tr>
<td>ADPREL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>ADPREH</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>ADACOL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>625</td>
</tr>
<tr>
<td>ADACOLH</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>624</td>
</tr>
<tr>
<td>ADACAP</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ADCAP&lt;4:0&gt;</td>
</tr>
<tr>
<td>ADRPT</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>626</td>
</tr>
<tr>
<td>ADCNT</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>627</td>
</tr>
<tr>
<td>ADFLTR</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>627</td>
</tr>
<tr>
<td>ADFLTRH</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>627</td>
</tr>
<tr>
<td>ADRESL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>628, 629</td>
</tr>
<tr>
<td>ADRESH</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>628, 629</td>
</tr>
<tr>
<td>ADPREVH</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>630</td>
</tr>
<tr>
<td>ADPREVL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>630</td>
</tr>
<tr>
<td>ADACCH</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>631</td>
</tr>
<tr>
<td>ADACCCL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ACC&lt;17:16&gt;</td>
<td>631</td>
</tr>
<tr>
<td>ADACCUC</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>631</td>
</tr>
<tr>
<td>ADSTPTL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>632</td>
</tr>
<tr>
<td>ADSTPHT</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>632</td>
</tr>
<tr>
<td>Name</td>
<td>Bit 7</td>
<td>Bit 6</td>
<td>Bit 5</td>
<td>Bit 4</td>
<td>Bit 3</td>
<td>Bit 2</td>
<td>Bit 1</td>
<td>Bit 0</td>
<td>Register on Page</td>
</tr>
<tr>
<td>----------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>------------------</td>
</tr>
<tr>
<td>ADERRL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ERR&lt;7:0&gt;</td>
<td>633</td>
</tr>
<tr>
<td>ADERRH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ERR&lt;15:8&gt;</td>
<td>633</td>
</tr>
<tr>
<td>ADLTHH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>LTH&lt;15:8&gt;</td>
<td>633</td>
</tr>
<tr>
<td>ADLTHL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>LTH&lt;7:0&gt;</td>
<td>634</td>
</tr>
<tr>
<td>ADUTHH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>UTH&lt;15:8&gt;</td>
<td>634</td>
</tr>
<tr>
<td>ADUTHL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>UTH&lt;7:0&gt;</td>
<td>634</td>
</tr>
<tr>
<td>ADERRL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ERR&lt;15:8&gt;</td>
<td>633</td>
</tr>
<tr>
<td>ADACT</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ADACT&lt;4:0&gt;</td>
<td>619</td>
</tr>
<tr>
<td>ADCP</td>
<td>CPON</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CPRDY</td>
<td>636</td>
</tr>
</tbody>
</table>

Legend: — = unimplemented read as ‘0’. Shaded cells are not used for the ADC module.
37.0 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 (VSOURCE+) of the DAC can be connected to:
- FVR Buffer
- External VREF+ pin
- VDD supply voltage

The negative input source (VSOURCE-) of the DAC can be connected to:
- External VREF- pin
- Vss

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 37-1: DIGITAL-TO-ANALOG CONVERTER BLOCK DIAGRAM

Note 1: The unbuffered DACx_output is provided on the DACxOUT pin(s).
37.1 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 37-1.

37.2 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 44-18.

37.3 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.

37.4 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.

37.5 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 37-1: DAC OUTPUT VOLTAGE

\[
IF \text{ DACEN} = 1
\]

\[
DAC_{\text{output}} = \left( (V_{\text{REF+}} - V_{\text{REF-}}) \times \frac{DATA[4:0]}{2^5} \right) + V_{\text{REF-}}
\]

Note: See the DAC1CON0 register for the available VSOURCE+ and VSOURCE- selections.
### 37.6 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.

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>Bit Name Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAC1</td>
<td>DAC1</td>
</tr>
</tbody>
</table>

#### REGISTER 37-1: DAC1CON0: DAC CONTROL REGISTER

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value 1</th>
<th>Value 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>EN</td>
<td>DAC Enable bit</td>
<td>DAC is enabled</td>
</tr>
<tr>
<td>6</td>
<td>OE1</td>
<td>DAC Voltage Output Enable bit</td>
<td>DAC voltage level is output on the DAC1OUT1 pin</td>
</tr>
<tr>
<td>5</td>
<td>OE2</td>
<td>DAC Voltage Output Enable bit</td>
<td>DAC voltage level is output on the DAC1OUT2 pin</td>
</tr>
<tr>
<td>4-3</td>
<td>PSS&lt;1:0&gt;</td>
<td>DAC Positive Source Select bit</td>
<td>Reserved</td>
</tr>
<tr>
<td>1</td>
<td>NSS</td>
<td>DAC Negative Source Select bit</td>
<td>VREF-</td>
</tr>
</tbody>
</table>

**Note 1:** DAC1OUTx output pins are still active.
REGISTER 37-2:  DAC1CON1: DAC DATA REGISTER

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>DATA&lt;4:0&gt;</td>
<td></td>
</tr>
</tbody>
</table>

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

bit 7-5 Unimplemented: Read as ‘0’
bit 4-0 **DATA<4:0>:** Data Input Register for DAC bits

TABLE 37-1:  SUMMARY OF REGISTERS ASSOCIATED WITH THE DAC MODULE

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAC1CON0</td>
<td>EN</td>
<td>—</td>
<td>OE1</td>
<td>OE2</td>
<td>PSS&lt;1:0&gt;</td>
<td>—</td>
<td>NSS</td>
<td>—</td>
<td>640</td>
</tr>
<tr>
<td>DAC1CON1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>DATA&lt;4:0&gt;</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td>641</td>
</tr>
</tbody>
</table>

Legend: — = Unimplemented location, read as ‘0’. Shaded cells are not used with the DAC module.
38.0 COMPARATOR MODULE

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

38.1 Comparator Overview

A single comparator is shown in Figure 38-1 along with the relationship between the analog input levels and the digital output. When the analog voltage at V\text{IN}^+ is less than the analog voltage at V\text{IN}^-, the output of the comparator is a digital low level. When the analog voltage at V\text{IN}^+ is greater than the analog voltage at V\text{IN}^-, the output of the comparator is a digital high level.

![Figure 38-1: SINGLE COMPARATOR](image)

Note: The black areas of the output of the comparator represents the uncertainty due to input offsets and response time.
Note 1: When CxON = 0, all multiplexer inputs are disconnected and the Comparator will produce a '0' at the output.
38.2 Comparator Control

Each comparator has two control registers: CMxCON0 and CMxCON1.

The CMxCON0 register (see Register 38-1) contains Control and Status bits for the following:

- Enable
- Output
- Output polarity
- Hysteresis enable
- Timer1 output synchronization

The CMxCON1 register (see Register 38-2) 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.

38.2.1 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.

38.2.2 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.

Table 38-1 shows the output state versus input conditions, including polarity control.

<table>
<thead>
<tr>
<th>Input Condition</th>
<th>POL</th>
<th>CxOUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>CxVN &gt; CxVP</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CxVN &lt; CxVP</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>CxVN &gt; CxVP</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>CxVN &lt; CxVP</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Note 1: The internal output of the comparator is latched with each instruction cycle. Unless otherwise specified, external outputs are not latched.
38.3 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 44-17 for more information.

38.3.1 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 38-2) and the Timer1 Block Diagram (Figure 21-1) for more information.

38.4 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
- CxE 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.

38.5 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 34.0 “Fixed Voltage Reference (FVR)” for more information on the Fixed Voltage Reference module.

See Section 37.0 “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.

38.6 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.
38.7 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 44-17 and Table 44-19 for more details.

38.8 Analog Input Connection Considerations

A simplified circuit for an analog input is shown in Figure 38-3. Since the analog input pins share their connection with a digital input, they have reverse biased ESD protection diodes to VDD and VSS. The analog input, therefore, must be between VSS and VDD. 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.

The maximum source impedance for analog sources is mentioned in Parameter AD08 in Table 44-15. 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.

Note 2: Analog levels on any pin defined as a digital input, may cause the input buffer to consume more current than is specified.

FIGURE 38-3: ANALOG INPUT MODEL

Legend: CPIN = Input Capacitance
ILEAKAGE = Leakage Current at the pin due to various junctions
RIC = Interconnect Resistance
RS = Source Impedance
VA = Analog Voltage
VT = Threshold Voltage

Note 1: See Section 44.0 “Electrical Specifications”.
38.9 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”).

38.10 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.

38.11 TMR2/4/6 Reset

The output of the comparator module can be used to reset Timer2. When the TxRST register is appropriately set, the timer will reset when the Comparator output goes high.

38.12 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.
38.13 Register Definitions: Comparator Control

Long bit name prefixes for the Comparators are shown in Table 38-2. Refer to Section 1.3.2.2 “Long Bit Names” for more information.

TABLE 38-2:

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>Bit Name Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>C1</td>
</tr>
<tr>
<td>C2</td>
<td>C2</td>
</tr>
</tbody>
</table>

REGISTER 38-1: CMxCON0: COMPARATOR x CONTROL REGISTER 0

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R-0/0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>U-0</th>
<th>U-1</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN</td>
<td>OUT</td>
<td>—</td>
<td>POL</td>
<td>—</td>
<td>—</td>
<td>HYS</td>
<td>SYNC</td>
</tr>
</tbody>
</table>

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 38-2: CMxCON1: COMPARATOR x CONTROL REGISTER 1

<table>
<thead>
<tr>
<th>bit 7-2</th>
<th>Unimplemented: Read as '0'</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 1</td>
<td><strong>INTP</strong>: Comparator Interrupt on Positive-Going Edge Enable bit</td>
</tr>
<tr>
<td>1</td>
<td>The CxIF interrupt flag will be set upon a positive-going edge of the CxOUT bit</td>
</tr>
<tr>
<td>0</td>
<td>No interrupt flag will be set on a positive-going edge of the CxOUT bit</td>
</tr>
<tr>
<td>bit 0</td>
<td><strong>INTN</strong>: Comparator Interrupt on Negative-Going Edge Enable bit</td>
</tr>
<tr>
<td>1</td>
<td>The CxIF interrupt flag will be set upon a negative-going edge of the CxOUT bit</td>
</tr>
<tr>
<td>0</td>
<td>No interrupt flag will be set on a negative-going edge of the CxOUT bit</td>
</tr>
</tbody>
</table>

### REGISTER 38-3: CMxNCH: COMPARATOR x INVERTING CHANNEL SELECT REGISTER

<table>
<thead>
<tr>
<th>bit 7-3</th>
<th>Unimplemented: Read as '0'</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 2-0</td>
<td><strong>NCH&lt;2:0&gt;</strong>: Comparator Inverting Input Channel Select bits</td>
</tr>
<tr>
<td>111</td>
<td>Vss</td>
</tr>
<tr>
<td>110</td>
<td>FVR_Buffer2</td>
</tr>
<tr>
<td>101</td>
<td>NCH not connected</td>
</tr>
<tr>
<td>100</td>
<td>NCH not connected</td>
</tr>
<tr>
<td>011</td>
<td>CxIN3-</td>
</tr>
<tr>
<td>010</td>
<td>CxIN2-</td>
</tr>
<tr>
<td>001</td>
<td>CxIN1-</td>
</tr>
<tr>
<td>000</td>
<td>CxIN0-</td>
</tr>
</tbody>
</table>

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

---

© 2017 Microchip Technology Inc. Preliminary
REGISTER 38-4: CMxPCH: COMPARATOR x NON-INVERTING CHANNEL SELECT REGISTER

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>PCH&lt;2:0&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 38-5: CMOUT: COMPARATOR OUTPUT REGISTER

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>R-0/0</th>
<th>R-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>R-0/0</td>
<td>R-0/0</td>
<td>C2OUT</td>
<td>C1OUT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 38-3: SUMMARY OF REGISTERS ASSOCIATED WITH COMPARATOR MODULE

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Reset Values on page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMxCON0</td>
<td>EN</td>
<td>OUT</td>
<td>—</td>
<td>POL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>648</td>
</tr>
<tr>
<td>CMxCON1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>649</td>
</tr>
<tr>
<td>CMxNCH</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>649</td>
</tr>
<tr>
<td>CMxPCH</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>650</td>
</tr>
<tr>
<td>CMOUT</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>650</td>
</tr>
</tbody>
</table>

Legend: — = unimplemented, read as '0'. Shaded cells are unused by the comparator module.
39.0 HIGH/LOW-VOLTAGE DETECT (HLVD)

The PIC18(L)F26/27/45/46/47/55/56/57K42 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 39-1. 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.
39.1 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 39-1: HLVD MODULE BLOCK DIAGRAM
39.2 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.

39.3 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 44-4).

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.

39.4 HLVD Start-up Time

The internal reference voltage of the HLVD module, specified in electrical specification (Table 44-19), 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, TFVRST, is an interval that is independent of device clock speed. It is specified in electrical specification (Table 44-19).

The HLVD interrupt flag is not enabled until TFVRST 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 39-2 or Figure 39-3).
FIGURE 39-2: LOW-VOLTAGE DETECT OPERATION (INTL = 1)

CASE 1:
- VDD
- HLVDIF
- Enable HLVD
- RDY

HLVDIF may not be Set

HLVDIF Cleared in Software

Band Gap Reference Voltage is Stable

CASE 2:
- VDD
- HLVDIF
- Enable HLVD
- RDY

Band Gap Reference Voltage is Stable

HLVDIF Cleared in Software

HLVDIF Cleared in Software.
HLVDIF Remains Set since HLVD Condition still Exists
39.5 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 39-4 shows a possible voltage curve. Over time, the device voltage decreases. When the device voltage reaches voltage, $V_A$, the HLVD logic generates an interrupt at time, $T_A$. 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 $T_B$. This would give the application a time window, represented by the difference between $T_A$ and $T_B$, to safely exit.
39.6 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.

39.7 Operation During Idle and Doze Modes

In both Idle and Doze modes, the module is active and events are generated if peripheral is enabled.

39.8 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.

39.9 Effects of a Reset

A device Reset forces all registers to their Reset state. This forces the HLVD module to be turned off.
39.10 Register Definitions: HLVD Control

Long bit name prefixes for the HLVD peripheral is shown in Table 39-1. Refer to Section 1.3.2.2 “Long Bit Names” for more information.

TABLE 39-1:

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>Bit Name Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>HLVD</td>
<td>HLVD</td>
</tr>
</tbody>
</table>

REGISTER 39-1: HLVDCON0: HIGH/LOW-VOLTAGE DETECT CONTROL REGISTER 0

<table>
<thead>
<tr>
<th></th>
<th>R/W-0/0</th>
<th>U-0</th>
<th>R-x</th>
<th>R-x</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN</td>
<td>—</td>
<td>—</td>
<td>OUT</td>
<td>RDY</td>
<td>—</td>
<td>—</td>
<td>INTH</td>
<td>INTL</td>
</tr>
<tr>
<td>bit 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 3-2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 ≤ selected detection limit (HLVDL<3:0>)
0 = Voltage ≥ 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 = HLVDF will be set when voltage ≥ selected detection limit (SEL<3:0>)
0 = HLVDF will not be set

bit 0  INTL: HLVD Negative going (Low Voltage) Interrupt Enable
1 = HLVDF will be set when voltage ≤ selected detection limit (SEL<3:0>)
0 = HLVDF will not be set
REGISTER 39-2: HLVDCON1: LOW-VOLTAGE DETECT CONTROL REGISTER 1

<table>
<thead>
<tr>
<th>bit 7-4</th>
<th>bit 3-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unimplemented: Read as ‘0’</td>
<td></td>
</tr>
</tbody>
</table>

bit 3-0 SEL<3:0>: High/Low Voltage Detection Limit Selection bits
Refer to Table 44-14 for voltage detection limits.

TABLE 39-2: SUMMARY OF REGISTERS ASSOCIATED WITH HIGH/LOW-VOLTAGE DETECT MODULE

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>HLVDCON0</td>
<td>EN</td>
<td>—</td>
<td>OUT</td>
<td>—</td>
<td>—</td>
<td>INTH</td>
<td>INTL</td>
<td></td>
<td>657</td>
</tr>
<tr>
<td>HLVDCON1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>SEL&lt;3:0&gt;</td>
<td></td>
<td>658</td>
</tr>
</tbody>
</table>

Legend: — = unimplemented, read as ‘0’. Shaded cells are unused by the HLVD module.
40.0 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 “PIC18F26/27/45/46/47/55/56/57K42 Memory Programming Specification” (DS40001886).

40.1 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 V\text{IH}.H.

40.2 Low-Voltage Programming Entry Mode

The Low-Voltage Programming Entry mode allows the PIC® Flash MCUs to be programmed using V\text{DD} 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. MCLR is brought to V\text{IL}.
2. A 32-bit key sequence is presented on ICSPDAT, while clocking ICSPCLK.

Once the key sequence is complete, MCLR must be held at V\text{IL} 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.

40.3 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 40-1.

FIGURE 40-1: 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 40-2.

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 40-3 for more information.

<table>
<thead>
<tr>
<th>Pin Description*</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 = VPP/MCLR</td>
</tr>
<tr>
<td>2 = VDD Target</td>
</tr>
<tr>
<td>3 = VSS (ground)</td>
</tr>
<tr>
<td>4 = ICSPDAT</td>
</tr>
<tr>
<td>5 = ICSPCLK</td>
</tr>
<tr>
<td>6 = No Connect</td>
</tr>
</tbody>
</table>
FIGURE 40-2: PICkit™ PROGRAMMER STYLE CONNECTOR INTERFACE

- Pin 1 Indicator
- Pin Description*
  1 = VPP/MCLR
  2 = Vdd Target
  3 = Vss (ground)
  4 = ICSPDAT
  5 = ICSPCLK
  6 = No Connect

* The 6-pin header (0.100" spacing) accepts 0.025" square pins.

FIGURE 40-3: TYPICAL CONNECTION FOR ICSP™ PROGRAMMING

- External Programming Signals
- VDD
- VPP
- VSS
- Data
- Clock
- Device to be Programmed
  - VDD
  - MCLR/VPP
  - Vss
  - ICSPDAT
  - ICSPCLK

* Isolation devices (as required).
41.0 INSTRUCTION SET SUMMARY

PIC18(L)F26/27/45/46/47/55/56/57K42 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.

41.1 Standard Instruction Set

The standard PIC18 instruction set adds many enhancements to the previous PIC® MCU instruction sets, while maintaining an easy migration from these PIC® 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 41-3 lists byte-oriented, bit-oriented, literal and control operations. Table 41-1 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 μs. If a conditional test is true, or the program counter is changed as a result of an instruction, the instruction execution time is 2 μs. Two-word branch instructions (if true) would take 3 μs.

Figure 41-1 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 41-3, lists the standard instructions recognized by the Microchip Assembler (MPASM®).

Section 41.1.1 “Standard Instruction Set” provides a description of each instruction.
TABLE 41-1: OPCODE FIELD DESCRIPTIONS

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| **a** | RAM access bit  
  a = 0: RAM location in Access RAM (BSR register is ignored)  
  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;  
  d = 0: store result in WREG,  
  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) |
| **fn** | FSR Number (0 to 2) |
| **fs** | 12-bit Register file address (000h to FFFh). This is the source address. |
| **fd** | 12-bit Register file address (000h to FFFh). This is the destination address. |
| **zs** | 7-bit literal offset for FSR2 to used as register file address (00h to FFFh). This is the source address. |
| **zd** | 7-bit literal offset for FSR2 to used as register file address (00h 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  
  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.  
  s = 0: do not update into/from shadow registers  
  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)  
  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 |
| **FCL** | Program Counter Low Byte |
| **FCH** | Program Counter High Byte |
| **PCLATBH** | Program Counter High Byte Latch |
| **PCLATLU** | 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 41-1: OPCODE FIELD DESCRIPTIONS (CONTINUED)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; &gt;</td>
<td>Register bit field</td>
</tr>
<tr>
<td>#</td>
<td>In the set of</td>
</tr>
<tr>
<td>italics</td>
<td>User defined term (font is courier)</td>
</tr>
</tbody>
</table>

FIGURE 41-1: General Format for Instructions (1/2)

**Byte-oriented** file register operations

<table>
<thead>
<tr>
<th>15</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPCODE</td>
<td>d</td>
<td>a</td>
<td>f (FILE #)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- d = 0 for result destination to be WREG register
- d = 1 for result destination to be file register (f)
- a = 0 to force Access Bank
- a = 1 for BSR to select bank
- f = 8-bit file register address

**Example Instruction**

ADDWF MYREG, W, B

**Byte to Byte** move operations (2-word)

<table>
<thead>
<tr>
<th>15</th>
<th>12</th>
<th>11</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPCODE</td>
<td>f (Source FILE #)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>12</th>
<th>11</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111</td>
<td>f (Destination FILE #)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- f = 12-bit file register address

**Example Instruction**

MOVFF MYREG1, MYREG2

**Byte to Byte** move operations (3-word)

<table>
<thead>
<tr>
<th>15</th>
<th>12</th>
<th>11</th>
<th>4</th>
<th>3</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPCODE</td>
<td>FILE #</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>12</th>
<th>11</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111</td>
<td>FILE #</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>12</th>
<th>11</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111</td>
<td>FILE #</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Example Instruction**

MOVFFL MYREG1, MYREG2

**Bit-oriented** file register operations

<table>
<thead>
<tr>
<th>15</th>
<th>12</th>
<th>11</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPCODE</td>
<td>b (BIT #)</td>
<td>a</td>
<td>f (FILE #)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- b = 3-bit position of bit in file register (f)
- a = 0 to force Access Bank
- a = 1 for BSR to select bank
- f = 8-bit file register address

**Example Instruction**

BSF MYREG, bit, B

**Literal** operations

<table>
<thead>
<tr>
<th>15</th>
<th>8</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPCODE</td>
<td>k (literal)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- k = 8-bit immediate value

**Example Instruction**

MOVLM 7Fh
**FIGURE 41-2: General Format for Instructions (2/2)**

### Control operations

**CALL, GOTO and Branch operations**

<table>
<thead>
<tr>
<th>15</th>
<th>8</th>
<th>7</th>
<th>0</th>
<th>GOTO Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>12</td>
<td>11</td>
<td>0</td>
<td>n &lt; 20-bit immediate value</td>
</tr>
<tr>
<td>1111</td>
<td>n &lt; 19:8 (literal)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**CALL MYFUNC**

<table>
<thead>
<tr>
<th>15</th>
<th>8</th>
<th>7</th>
<th>0</th>
<th>CALL MYFUNC</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>12</td>
<td>11</td>
<td>0</td>
<td>n &lt; 20-bit immediate value</td>
</tr>
<tr>
<td>1111</td>
<td>n &lt; 19:8 (literal)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**S = Fast bit**

<table>
<thead>
<tr>
<th>15</th>
<th>11</th>
<th>10</th>
<th>0</th>
<th>BRA MYFUNC</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>8</td>
<td>7</td>
<td>0</td>
<td>n &lt; 10:0 (literal)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>8</th>
<th>7</th>
<th>0</th>
<th>BC MYFUNC</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>8</td>
<td>7</td>
<td>0</td>
<td>n &lt; 7:0 (literal)</td>
</tr>
</tbody>
</table>
### TABLE 41-2: INSTRUCTION SET

<table>
<thead>
<tr>
<th>Mnemonic, Operands</th>
<th>Description</th>
<th>Cycles</th>
<th>16-Bit Instruction Word</th>
<th>Status Affected</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>MSb</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LSb</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>BYTE-ORIENTED FILE REGISTER INSTRUCTIONS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADDWF f, d, a</td>
<td>Add WREG and f</td>
<td>1</td>
<td>0010 01da ffff ffff</td>
<td>C, DC, Z, OV, N</td>
<td></td>
</tr>
<tr>
<td>ADDWFC f, d, a</td>
<td>Add WREG and Carry bit to f</td>
<td>1</td>
<td>0010 00da ffff ffff</td>
<td>C, DC, Z, OV, N</td>
<td></td>
</tr>
<tr>
<td>ANDWF f, d, a</td>
<td>AND WREG with f</td>
<td>1</td>
<td>0001 01da ffff ffff</td>
<td>Z, N</td>
<td></td>
</tr>
<tr>
<td>CLRF f, a</td>
<td>Clear f</td>
<td>1</td>
<td>0110 101a ffff ffff</td>
<td>Z</td>
<td></td>
</tr>
<tr>
<td>COMF f, d, a</td>
<td>Complement f</td>
<td>1</td>
<td>0001 11da ffff ffff</td>
<td>Z, N</td>
<td></td>
</tr>
<tr>
<td>DECF f, d, a</td>
<td>Decrement f</td>
<td>1</td>
<td>0000 01da ffff ffff</td>
<td>C, DC, Z, OV, N</td>
<td></td>
</tr>
<tr>
<td>INC f, d, a</td>
<td>Increment f</td>
<td>1</td>
<td>0010 10da ffff ffff</td>
<td>C, DC, Z, OV, N</td>
<td></td>
</tr>
<tr>
<td>IORWF f, d, a</td>
<td>Inclusive OR WREG with f</td>
<td>1</td>
<td>0001 00da ffff ffff</td>
<td>Z, N</td>
<td></td>
</tr>
<tr>
<td>MOVF f, d, a</td>
<td>Move f to WREG or f</td>
<td>1</td>
<td>0101 00da ffff ffff</td>
<td>Z, N</td>
<td></td>
</tr>
<tr>
<td>MOVFF f, d</td>
<td>Move f (source) to 1st word</td>
<td>2</td>
<td>1100 ffff ffff ffff</td>
<td>None</td>
<td>2, 3</td>
</tr>
<tr>
<td>MOVFF f, d</td>
<td>f (destination) 2nd word</td>
<td>2</td>
<td>1111 ffff ffff ffff</td>
<td>None</td>
<td>3</td>
</tr>
<tr>
<td>MOVFF f, d</td>
<td>g (full destination) 3rd word</td>
<td>3</td>
<td>1111 ffff ffff ffff</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>MOVFF f, d</td>
<td>f (full destination)</td>
<td>3</td>
<td>1111 gggg gggg gggg</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>MOVWF f, a</td>
<td>Move WREG to f</td>
<td>1</td>
<td>0110 111a ffff ffff</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>MULWF f, a</td>
<td>Multiply WREG with f</td>
<td>1</td>
<td>0000 001a ffff ffff</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>NEG f, a</td>
<td>Negate f</td>
<td>1</td>
<td>0110 110a ffff ffff</td>
<td>C, DC, Z, OV, N</td>
<td></td>
</tr>
<tr>
<td>RLCF f, d, a</td>
<td>Rotate Left f through Carry</td>
<td>1</td>
<td>0011 01da ffff ffff</td>
<td>C, Z, N</td>
<td></td>
</tr>
<tr>
<td>RLNCF f, d, a</td>
<td>Rotate Left f (No Carry)</td>
<td>1</td>
<td>0100 01da ffff ffff</td>
<td>Z, N</td>
<td></td>
</tr>
<tr>
<td>RRCF f, d, a</td>
<td>Rotate Right f through Carry</td>
<td>1</td>
<td>0011 00da ffff ffff</td>
<td>C, Z, N</td>
<td></td>
</tr>
<tr>
<td>RRNCF f, d, a</td>
<td>Rotate Right f (No Carry)</td>
<td>1</td>
<td>0100 00da ffff ffff</td>
<td>Z, N</td>
<td></td>
</tr>
<tr>
<td>SET f, a</td>
<td>Set f</td>
<td>1</td>
<td>0110 100a ffff ffff</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>SUBF WB f, d, a</td>
<td>Subtract f from WREG with borrow</td>
<td>1</td>
<td>0101 01da ffff ffff</td>
<td>C, DC, Z, OV, N</td>
<td></td>
</tr>
<tr>
<td>SUBWF f, d, a</td>
<td>Subtract WREG from f</td>
<td>1</td>
<td>0101 11da ffff ffff</td>
<td>C, DC, Z, OV, N</td>
<td></td>
</tr>
<tr>
<td>SUBWF f, d, a</td>
<td>Subtract WREG from f with borrow</td>
<td>1</td>
<td>0101 10da ffff ffff</td>
<td>C, DC, Z, OV, N</td>
<td></td>
</tr>
<tr>
<td>SWAPF f, d, a</td>
<td>Swap nibbles in f</td>
<td>1</td>
<td>0011 10da ffff ffff</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>XORWF f, d, a</td>
<td>Exclusive OR WREG with f</td>
<td>1</td>
<td>0001 10da ffff ffff</td>
<td>Z, N</td>
<td></td>
</tr>
<tr>
<td><strong>BYTE-ORIENTED SKIP INSTRUCTIONS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPFSEQ f, a</td>
<td>Compare f with WREG, skip = 0</td>
<td>1</td>
<td>0110 001a ffff ffff</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>CPFSGT f, a</td>
<td>Compare f with WREG, skip &gt; 0</td>
<td>1</td>
<td>0110 010a ffff ffff</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>CPFSLT f, a</td>
<td>Compare f with WREG, skip &lt; 0</td>
<td>1</td>
<td>0110 000a ffff ffff</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>DECFSZ f, d, a</td>
<td>Decrement f, Skip if 0</td>
<td>1</td>
<td>0010 111a ffff ffff</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>DCFSNZ f, d, a</td>
<td>Decrement f, Skip if Not 0</td>
<td>1</td>
<td>0100 111a ffff ffff</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>INCFSZ f, d, a</td>
<td>Increment f, Skip if 0</td>
<td>1</td>
<td>0011 111a ffff ffff</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>INFSNZ f, d, a</td>
<td>Increment f, Skip if Not 0</td>
<td>1</td>
<td>0100 110a ffff ffff</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>TSTFSZ f, a</td>
<td>Test f, skip if 0</td>
<td>1</td>
<td>0110 011a ffff ffff</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td><strong>BIT-ORIENTED FILE REGISTER INSTRUCTIONS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCF f, b, a</td>
<td>Bit Clear f</td>
<td>1</td>
<td>1001 bbba ffff ffff</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>BSF f, b, a</td>
<td>Bit Set f</td>
<td>1</td>
<td>1000 bbba ffff ffff</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>BTG f, d, a</td>
<td>Bit Toggle f</td>
<td>1</td>
<td>0111 bbba ffff ffff</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td><strong>BIT-ORIENTED SKIP INSTRUCTIONS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BTFS f, b, a</td>
<td>Bit Test f, Skip if Clear</td>
<td>1</td>
<td>1011 bbba ffff ffff</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>BTFFS f, b, a</td>
<td>Bit Test f, Skip if Set</td>
<td>1</td>
<td>1010 bbba ffff ffff</td>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

**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**.

**Note 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.

**Note 3:** \( f, d, a \) and \( f, d, a \) 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 41-2: INSTRUCTION SET (CONTINUED)

<table>
<thead>
<tr>
<th>Mnemonic, Operands</th>
<th>Description</th>
<th>Cycles</th>
<th>16-Bit Instruction Word</th>
<th>Status Affected</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>MSb</td>
<td>LSb</td>
<td></td>
</tr>
<tr>
<td><strong>CONTROL INSTRUCTIONS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BC n</td>
<td>Branch if Carry</td>
<td>1 (2)</td>
<td>1110</td>
<td>0010</td>
<td>nnnn</td>
</tr>
<tr>
<td>BN n</td>
<td>Branch if Negative</td>
<td>1 (2)</td>
<td>1110</td>
<td>0110</td>
<td>nnnn</td>
</tr>
<tr>
<td>BNC n</td>
<td>Branch if Not Carry</td>
<td>1 (2)</td>
<td>1110</td>
<td>0011</td>
<td>nnnn</td>
</tr>
<tr>
<td>BNN n</td>
<td>Branch if Not Negative</td>
<td>1 (2)</td>
<td>1110</td>
<td>0111</td>
<td>nnnn</td>
</tr>
<tr>
<td>BNOV n</td>
<td>Branch if Not Overflow</td>
<td>1 (2)</td>
<td>1110</td>
<td>0101</td>
<td>nnnn</td>
</tr>
<tr>
<td>BNZ n</td>
<td>Branch if Not Zero</td>
<td>2</td>
<td>1110</td>
<td>0001</td>
<td>nnnn</td>
</tr>
<tr>
<td>BOV n</td>
<td>Branch if Overflow</td>
<td>1 (2)</td>
<td>1110</td>
<td>0100</td>
<td>nnnn</td>
</tr>
<tr>
<td>BRA n</td>
<td>Branch Unconditionally</td>
<td>1 (2)</td>
<td>1101</td>
<td>0nnn</td>
<td>nnnn</td>
</tr>
<tr>
<td>BZ n</td>
<td>Branch if Zero</td>
<td>1 (2)</td>
<td>1110</td>
<td>0000</td>
<td>nnnn</td>
</tr>
<tr>
<td>CALL n, s</td>
<td>Call subroutine 1st word</td>
<td>1 (2)</td>
<td>1110</td>
<td>110s</td>
<td>nnnn</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CALLW —</td>
<td>W -&gt; PCL and Call subroutine</td>
<td>2</td>
<td>1110</td>
<td>1111</td>
<td>nnnn</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RCALL n</td>
<td>Relative Call</td>
<td>2</td>
<td>1101</td>
<td>1nnn</td>
<td>nnnn</td>
</tr>
<tr>
<td>RETFIE s</td>
<td>Return from interrupt enable</td>
<td>2</td>
<td>0000</td>
<td>0000</td>
<td>0001</td>
</tr>
<tr>
<td>RETLW k</td>
<td>Return with literal in WREG</td>
<td>2</td>
<td>0000</td>
<td>1100</td>
<td>kkkk</td>
</tr>
<tr>
<td>RETURN s</td>
<td>Return from Subroutine</td>
<td>2</td>
<td>0000</td>
<td>0000</td>
<td>0001</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>INHERENT INSTRUCTIONS</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CLEARWDT —</td>
<td>Clear Watchdog Timer</td>
<td>1</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>DAW —</td>
<td>Decimal Adjust WREG</td>
<td>1</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>NOP —</td>
<td>No Operation</td>
<td>1</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>NOP —</td>
<td>No Operation</td>
<td>1</td>
<td>1111</td>
<td>xxxx</td>
<td>xxxx</td>
</tr>
<tr>
<td>POP —</td>
<td>Pop top of return stack (TOS)</td>
<td>1</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>PUSH —</td>
<td>Push top of return stack (TOS)</td>
<td>1</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>RESET</td>
<td>Software device Reset</td>
<td>1</td>
<td>0000</td>
<td>0000</td>
<td>1111</td>
</tr>
<tr>
<td>SLEEP</td>
<td>Go into Standby mode</td>
<td>1</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
</tbody>
</table>

**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**.

**Note 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.

**Note 3:** $f_n$ 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 41-2: INSTRUCTION SET (CONTINUED)

<table>
<thead>
<tr>
<th>Mnemonic, Operands</th>
<th>Description</th>
<th>Cycles</th>
<th>16-Bit Instruction Word</th>
<th>Status Affected</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>MSb</td>
<td>LSb</td>
<td></td>
</tr>
<tr>
<td><strong>LITERAL INSTRUCTIONS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADDLW k</td>
<td>Add literal and WREG</td>
<td>1</td>
<td>0000 1111 kkkk kkkk</td>
<td>C, DC, Z, OV, N</td>
<td></td>
</tr>
<tr>
<td>ANDLW k</td>
<td>AND literal with WREG</td>
<td>1</td>
<td>0000 1011 kkkk kkkk</td>
<td>Z, N</td>
<td></td>
</tr>
<tr>
<td>IORLW k</td>
<td>Inclusive OR literal with WREG</td>
<td>1</td>
<td>0000 1001 kkkk kkkk</td>
<td>Z, N</td>
<td></td>
</tr>
<tr>
<td>LFSR fn, k</td>
<td>Load FSR(fn) with a 14-bit literal (k)</td>
<td>2</td>
<td>1110 1110 00ff kkkk</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1111 00kk kkkk kkkk</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADDFSR fn, k</td>
<td>Add FSR(fn) with (k)</td>
<td>1</td>
<td>1110 1000 ffkk kkkk</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>SUBFSR fn, k</td>
<td>Subtract (k) from FSR(fn)</td>
<td>1</td>
<td>1110 1001 ffkk kkkk</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>MOVLB k</td>
<td>Move literal to BSR&lt;5:0&gt;</td>
<td>1</td>
<td>0000 0011 00kk kkkk</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>MOVLR k</td>
<td>Move literal to WREG</td>
<td>1</td>
<td>0000 1110 kkkk kkkk</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>MULLW k</td>
<td>Multiply literal with WREG</td>
<td>1</td>
<td>0000 1101 kkkk kkkk</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>RETLW k</td>
<td>Return with literal in WREG</td>
<td>2</td>
<td>0000 1100 kkkk kkkk</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>SUBLR k</td>
<td>Subtract WREG from literal</td>
<td>1</td>
<td>0000 1000 kkkk kkkk</td>
<td>C, DC, Z, OV, N</td>
<td></td>
</tr>
<tr>
<td>XORLW k</td>
<td>Exclusive OR literal with WREG</td>
<td>1</td>
<td>0000 1010 kkkk kkkk</td>
<td>Z, N</td>
<td></td>
</tr>
</tbody>
</table>

**DATA MEMORY – PROGRAM MEMORY INSTRUCTIONS**

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Description</th>
<th>Cycles</th>
<th>16-Bit Instruction Word</th>
<th>Status Affected</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBLRD*</td>
<td>Table Read</td>
<td>2 - 5</td>
<td>0000 0000 0000 1000</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>TBLRD*+</td>
<td>Table Read with post-increment</td>
<td>2 - 5</td>
<td>0000 0000 0000 1001</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>TBLRD*-</td>
<td>Table Read with post-decrement</td>
<td>2 - 5</td>
<td>0000 0000 0000 1010</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>TBLRD+*</td>
<td>Table Read with pre-increment</td>
<td>2 - 5</td>
<td>0000 0000 0000 1011</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>TBLWT*</td>
<td>Table Write</td>
<td>2 - 5</td>
<td>0000 0000 0000 1100</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>TBLWT*+</td>
<td>Table Write with post-increment</td>
<td>2 - 5</td>
<td>0000 0000 0000 1101</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>TBLWT*-</td>
<td>Table Write with post-decrement</td>
<td>2 - 5</td>
<td>0000 0000 0000 1110</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>TBLWT+*</td>
<td>Table Write with pre-increment</td>
<td>2 - 5</td>
<td>0000 0000 0000 1111</td>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

**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.

**Note 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.

**Note 3:** fn and fd 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.
41.1.1 STANDARD INSTRUCTION SET

ADDFSR  Add Literal to FSR

Syntax: ADDFSR f, k
Operands: 0 ≤ k ≤ 63
          f ∈ { 0, 1, 2 }
Operation: FSR(f) + k → 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:

Example: ADDFSR 2, 23h
Before Instruction
FSR2 = 03FFh
After Instruction
FSR2 = 0422h

ADDLW  ADD literal to W

Syntax: ADDLW k
Operands: 0 ≤ k ≤ 255
Operation: (W) + k → 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:

Example: ADDLW 15h
Before Instruction
W = 10h
After Instruction
W = 25h

ADDWF  ADD W to f

Syntax: ADDWF f {,d {,a}}
Operands: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]
Operation: (W) + (f) → 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 ≤ 95 (5Fh). See Section 41.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.
Words: 1
Cycles: 1
Q Cycle Activity:

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**

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>ADDWFC f{,d{,a}}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands:</td>
<td>0 ≤ f ≤ 255</td>
</tr>
<tr>
<td></td>
<td>d ∈ [0,1]</td>
</tr>
<tr>
<td></td>
<td>a ∈ [0,1]</td>
</tr>
<tr>
<td>Operation:</td>
<td>(W) + (f) + (C) → dest</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>N,OV,C,DC,Z</td>
</tr>
<tr>
<td>Encoding:</td>
<td>0010 00da ffff ffff</td>
</tr>
<tr>
<td>Description:</td>
<td>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. 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 Section 41.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.</td>
</tr>
<tr>
<td>Words:</td>
<td>1</td>
</tr>
<tr>
<td>Cycles:</td>
<td>1</td>
</tr>
</tbody>
</table>

**Example:**

<table>
<thead>
<tr>
<th>Decode</th>
<th>Read register 'f'</th>
<th>Process Data</th>
<th>Write to destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>Q2</td>
<td>Q3</td>
<td>Q4</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>ANDLW k</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands:</td>
<td>0 ≤ k ≤ 255</td>
</tr>
<tr>
<td>Operation:</td>
<td>(W) .AND. k → W</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>N, Z</td>
</tr>
<tr>
<td>Encoding:</td>
<td>0000 1011 kkkk kkkk</td>
</tr>
<tr>
<td>Description:</td>
<td>The contents of W are AND’ed with the 8-bit literal 'k'. The result is placed in W.</td>
</tr>
<tr>
<td>Words:</td>
<td>1</td>
</tr>
<tr>
<td>Cycles:</td>
<td>1</td>
</tr>
</tbody>
</table>

**Q Cycle Activity:**

<table>
<thead>
<tr>
<th>Decode</th>
<th>Read literal 'k'</th>
<th>Process Data</th>
<th>Write to W</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>Q2</td>
<td>Q3</td>
<td>Q4</td>
</tr>
</tbody>
</table>

Example: ANDLW 05Fh

Before Instruction
- W = A3h

After Instruction
- W = 03h

---

© 2017 Microchip Technology Inc. Preliminary DS40001919B-page 669
ANDWF AND W with f

Syntax: ANDWF f,d {a}

Operands:
- \( f \in [0, 255] \)
- \( d \in [0, 1] \)
- \( a \in [0, 1] \)

Operation: \((W) \land (f) \rightarrow \text{dest}\)

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 41.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.

Words: 1
Cycles: 1

Q Cycle Activity:

Example:
```
Before Instruction
\( W = 17h \)
\( \text{REG} = C2h \)
After Instruction
\( W = 02h \)
\( \text{REG} = C2h \)
```

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:

Example:
```
HERE BC 5
Before Instruction
\( \text{PC} = \text{address (HERE)} \)
After Instruction
If CARRY = 1; \( \text{PC} = \text{address (HERE + 12)} \)
If CARRY = 0; \( \text{PC} = \text{address (HERE + 2)} \)
```
BCF Bit Clear f

Syntax:
BCF f, b {,a}

Operands:
0 ≤ f ≤ 255
0 ≤ b ≤ 7
a ∈ [0,1]

Operation:
0 → f<b>

Status Affected: None

Encoding:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

Description:
Bit 'b' in register 'f' is cleared.
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 Section 41.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.

Words: 1
Cycles: 1

Q Cycle Activity:

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 ≤ n ≤ 127

Operation:
if NEGATIVE bit is '1'
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1110</td>
<td>0110</td>
<td>nnnn</td>
<td>nnnn</td>
</tr>
</tbody>
</table>

Description:
If the NEGATIVE 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:

Example:
HERE BN Jump

Before Instruction
PC = address (HERE)
After Instruction
If NEGATIVE = 1;
PC = address (Jump)
If NEGATIVE = 0;
PC = address (HERE + 2)
BNC  Branch if Not Carry
Syntax:  BNC n
Operands: -128 ≤ n ≤ 127
Operation: if CARRY bit is ‘0’
(PC) + 2 + 2n → PC
Status Affected: None
Encoding:  1110 0011 nnnn nnnn
Description: If the CARRY 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:
Decide  Read literal
Q1    Q2    Q3    Q4
' n'    Process     Write to PC
No operation  No      No          No
operation
If No Jump:
Decide  Read literal
Q1    Q2    Q3    Q4
' n'    Process     No
No operation  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 ≤ n ≤ 127
Operation: if NEGATIVE bit is ‘0’
(PC) + 2 + 2n → PC
Status Affected: None
Encoding:  1110 0111 nnnn nnnn
Description: If the NEGATIVE 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:
Decide  Read literal
Q1    Q2    Q3    Q4
' n'    Process     Write to PC
No operation  No      No          No
operation
If No Jump:
Decide  Read literal
Q1    Q2    Q3    Q4
' n'    Process     No
No operation  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)
### BNOV  Branch if Not Overflow

**Syntax:**

\[
\text{BNOV } n
\]

**Operands:**

\[-128 \leq n \leq 127\]

**Operation:**

If OVERFLOW bit is ‘0’

\[
(PC) + 2 + 2n \rightarrow PC
\]

**Status Affected:** None

**Encoding:**

\[
\begin{array}{cccc}
1110 & 0101 & \text{nnnn} & \text{nnnn}
\end{array}
\]

**Description:**

If the OVERFLOW bit is ‘0’, 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:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read literal</td>
<td>Process Data</td>
<td>Write to PC</td>
</tr>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

**If No Jump:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read literal</td>
<td>Process Data</td>
<td>No operation</td>
</tr>
<tr>
<td>'n'</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

**Example:**

Before Instruction

\[
\text{PC} = \text{address (HERE)}
\]

After Instruction

If OVERFLOW = 0;

\[
\text{PC} = \text{address (Jump)}
\]

If OVERFLOW = 1;

\[
\text{PC} = \text{address (HERE + 2)}
\]

### BNZ  Branch if Not Zero

**Syntax:**

\[
\text{BNZ } n
\]

**Operands:**

\[-128 \leq n \leq 127\]

**Operation:**

If ZERO bit is ‘0’

\[
(PC) + 2 + 2n \rightarrow PC
\]

**Status Affected:** None

**Encoding:**

\[
\begin{array}{cccc}
1110 & 0001 & \text{nnnn} & \text{nnnn}
\end{array}
\]

**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:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read literal</td>
<td>Process Data</td>
<td>Write to PC</td>
</tr>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

**If No Jump:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read literal</td>
<td>Process Data</td>
<td>No operation</td>
</tr>
<tr>
<td>'n'</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

**Example:**

Before Instruction

\[
\text{PC} = \text{address (HERE)}
\]

After Instruction

If ZERO = 0;

\[
\text{PC} = \text{address (Jump)}
\]

If ZERO = 1;

\[
\text{PC} = \text{address (HERE + 2)}
\]
**BRA**  
Unconditional Branch

Syntax: \( \text{BRA } n \)

Operands: \(-1024 \leq n \leq 1023\)

Operation: \((\text{PC}) + 2 + 2n \rightarrow \text{PC}\)

Status Affected: None

Encoding: \(1101 \ 0\text{nnn} \ \text{nnnn} \ \text{nnnn}\)

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 \(\text{PC} + 2 + 2n\). This instruction is a 2-cycle instruction.

Words: 1

Cycles: 2

Q Cycle Activity:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read literal 'n'</td>
<td>Process Data</td>
<td>Write to PC</td>
</tr>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

Example:

Before Instruction
PC = address (HERE)

After Instruction
PC = address (Jump)

**BSF**  
Bit Set f

Syntax: \( \text{BSF } f, b \ (a) \)

Operands: 
- \(0 \leq f \leq 255\)
- \(0 \leq b \leq 7\)
- \(a \in [0,1]\)

Operation: \(1 \rightarrow f < b>\)

Status Affected: None

Encoding: \(1000 \ b\text{ba} \ f\text{fff} \ f\text{fff}\)

Description: Bit ‘b’ in register ‘f’ is set. 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 41.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.

Words: 1

Cycles: 1

Q Cycle Activity:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read register 'f'</td>
<td>Process Data</td>
<td>Write register 'f'</td>
</tr>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

Example:

Before Instruction
FLAG_REG = 0Ah

After Instruction
FLAG_REG = 8Ah
**BTFSC**  
**Bit Test File, Skip if Clear**

**Syntax:**  
BTFSC  f, b {,a}

**Operands:**  
0 ≤ f ≤ 255  
0 ≤ b ≤ 7  
a ∈ [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 ≤ 95 (5Fh).

See Section 41.2.3 “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:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read</td>
<td>Process</td>
<td>Data</td>
</tr>
</tbody>
</table>

If skip:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

If skip and followed by 2-word instruction:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

**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 ≤ f ≤ 255  
0 ≤ b ≤ 7  
a ∈ [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 ≤ 95 (5Fh).

See Section 41.2.3 “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:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read</td>
<td>Process</td>
<td>Data</td>
</tr>
</tbody>
</table>

If skip:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

If skip and followed by 2-word instruction:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>BTG f, b {,a}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands:</td>
<td>$0 \leq f \leq 255$</td>
</tr>
<tr>
<td></td>
<td>$0 \leq b &lt; 7$</td>
</tr>
<tr>
<td></td>
<td>$a \in [0,1]$</td>
</tr>
<tr>
<td>Operation:</td>
<td>$(f&lt;<em>{b}&gt; \rightarrow f&lt;</em>{b}&gt;)$</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>None</td>
</tr>
<tr>
<td>Encoding:</td>
<td>0111 bbba ffff ffff</td>
</tr>
<tr>
<td>Description:</td>
<td>Bit 'b' in data memory location 'f' is inverted. 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 41.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.</td>
</tr>
</tbody>
</table>

Words: 1
Cycles: 1

Q Cycle Activity:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read register 'f'</td>
<td>Process Data</td>
<td>Write register 'f'</td>
</tr>
</tbody>
</table>

Example: BTG PORTC, 4, 0
Before Instruction: PORTC = 0111 0101 [75h]
After Instruction: PORTC = 0110 0101 [65h]

BOV Branch if Overflow

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>BOV n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands:</td>
<td>$-128 \leq n \leq 127$</td>
</tr>
<tr>
<td>Operation:</td>
<td>if OVERFLOW bit is '1' ( (PC) + 2 + 2n \rightarrow PC )</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>None</td>
</tr>
<tr>
<td>Encoding:</td>
<td>1110 0100 nnnn nnnn</td>
</tr>
<tr>
<td>Description:</td>
<td>If the OVERFLOW 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.</td>
</tr>
</tbody>
</table>

Words: 1
Cycles: 1(2)

Q Cycle Activity:

If Jump:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read literal 'n'</td>
<td>Process Data</td>
<td>Write to PC</td>
</tr>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

If No Jump:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read literal 'n'</td>
<td>Process Data</td>
<td>No operation</td>
</tr>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

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’  
(PC) + 2 + 2n → PC

**Status Affected:**  
None

**Encoding:**  
```
1110 0000 nnnn nnnn
```

**Description:**  
If the ZERO 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:**  

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read literal</td>
<td>Process Data</td>
<td>Write to PC</td>
</tr>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read literal</td>
<td>Process Data</td>
<td>No operation</td>
</tr>
<tr>
<td>'n'</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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 ≤ k ≤ 1048575  
s ∈ [0,1]

**Operation:**  
(PC) + 4 → TOS,  
k → PC<20:1>,  
if s = 1  
(W) → WS,  
(Status) → STATUSS,  
(BSR) → BSRS

**Status Affected:**  
None

**Encoding:**  
```
1110 1111 110s 1111 k19kkk kkkk kkkk0 kkkk8
```

**Description:**  
Subroutine call of entire 2-Mbyte memory range. First, return address (PC + 4) is pushed onto the return stack. If ‘s’ = 1, the W, Status and BSR registers are also pushed into their respective shadow registers, WS, STATUSS and BSRS. If ‘s’ = 0, no update occurs (default). Then, the 20-bit value ‘k’ is loaded into PC<20:1>. CALL is a 2-cycle instruction.

**Words:**  
2

**Cycles:**  
2

**Q Cycle Activity:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read literal</td>
<td>PUSH PC to stack</td>
<td>Read literal</td>
</tr>
<tr>
<td>'k'&lt;7:0&gt;,</td>
<td></td>
<td></td>
<td>'k'&lt;19:8&gt;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Write to PC</td>
</tr>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

**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,
(W) → PCL,
(PCLATH) → PCH,
(PCLATU) → PCU
Status Affected: None
Encoding: 0000 0000 0001 0100
Description: 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. Unlike CALL, there is no option to update W, Status or BSR.

Words: 1
Cycles: 2
Q Cycle Activity:

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
a ∈ [0,1]
Operation: 000h → f
1 → Z
Status Affected: Z
Encoding: 0110 101a ffff ffff
Description: Clears the contents of the specified register. 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 Section 41.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.

Words: 1
Cycles: 1
Q Cycle Activity:

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` → TO,
- `1` → PD

**Status Affected:** TO, PD

**Encoding:**

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0100</td>
<td></td>
</tr>
</tbody>
</table>

**Description:**

CLRWDT instruction resets the Watchdog Timer. It also resets the post-scaler of the WDT. Status bits, TO and PD, are set.

**Words:** 1

**Cycles:** 1

**Q Cycle Activity:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>No operation</td>
<td>Process Data</td>
<td>No operation</td>
</tr>
</tbody>
</table>

**Example:**

- **Before Instruction**
  - WDT Counter = ?

- **After Instruction**
  - WDT Counter = `00h`
  - WDT Postscaler = `0`
  - TO = `1`
  - PD = `1`

### COMF

**Complement f**

**Syntax:** COMF `f,d{,a}`

**Operands:**
- `0 ≤ f ≤ 255`
- `d ∈ [0,1]`
- `a ∈ [0,1]`

**Operation:**
- `(f) → dest`

**Status Affected:** N, Z

**Encoding:**

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>lida</td>
<td>ffff</td>
<td>ffff</td>
<td></td>
</tr>
</tbody>
</table>

**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 ≤ 95 (5Fh)`. See Section 41.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.

**Words:** 1

**Cycles:** 1

**Q Cycle Activity:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read register 'f'</td>
<td>Process Data</td>
<td>Write to destination</td>
</tr>
</tbody>
</table>

**Example:**

- **Before Instruction**
  - REG = `13h`

- **After Instruction**
  - REG = `13h`
  - W = ECh
**CPFSEQ**  
**Compare f with W, skip if f = W**

**Syntax:** CPFSEQ  \( f \{,a \} \)

**Operands:**
- \( 0 \leq f \leq 255 \)
- \( a \in [0,1] \)

**Operation:**
- \( (f) - (W), \) skip if \( (f) = (W) \)  
  (unsigned comparison)

**Status Affected:** None

**Encoding:**
- 0110 001a 0000 0000

**Description:**
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 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 41.2.3 “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:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read register 'f'</td>
<td>Process Data</td>
<td>No operation</td>
</tr>
<tr>
<td>If skip:</td>
<td>Q1</td>
<td>Q2</td>
<td>Q3</td>
</tr>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
<tr>
<td>If skip and followed by 2-word instruction:</td>
<td>Q1</td>
<td>Q2</td>
<td>Q3</td>
</tr>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

**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:**
- \( 0 \leq f \leq 255 \)
- \( a \in [0,1] \)

**Operation:**
- \( (f) - (W), \) skip if \( (f) > (W) \)  
  (unsigned comparison)

**Status Affected:** None

**Encoding:**
- 0110 010a 0000 0000

**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 greater than the contents of WREG, then the fetched instruction 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 41.2.3 “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:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read register 'f'</td>
<td>Process Data</td>
<td>No operation</td>
</tr>
<tr>
<td>If skip:</td>
<td>Q1</td>
<td>Q2</td>
<td>Q3</td>
</tr>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
<tr>
<td>If skip and followed by 2-word instruction:</td>
<td>Q1</td>
<td>Q2</td>
<td>Q3</td>
</tr>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

**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 ≤ f ≤ 255
a ∈ [0,1]
Operation: (f) – (W), skip if (f) < (W)
(unsigned comparison)
Status Affected: None
Encoding:
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.
Words: 1
Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

DAW Decimal Adjust W Register

Syntax: DAW
Operands: None
Operation: If [W<3:0> > 9] or [DC = 1] then
(W<3:0>) + 6 → W<3:0>;
else
(W<3:0>) → W<3:0>;
If [W<7:4> + DC > 9] or [C = 1] then
(W<7:4>) + 6 + DC → W<7:4> ;
else
(W<7:4>) + DC → W<7:4>
Status Affected: C
Encoding: 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.

Q Cycle Activity:

Example 1:
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

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)
### 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:**

\[
\begin{array}{cccc}
\text{0000} & \text{01da} & \text{fff} & \text{fff}
\end{array}
\]

**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 41.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.

**Words:** 1

**Cycles:** 1

**Q Cycle Activity:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read register 'f'</td>
<td>Process Data</td>
<td>Write to destination</td>
</tr>
</tbody>
</table>

**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:**
\((f) - 1 \rightarrow \text{dest}, \text{skip if result = 0}\)

**Status Affected:** None

**Encoding:**

\[
\begin{array}{cccc}
\text{0010} & \text{11da} & \text{fff} & \text{fff}
\end{array}
\]

**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 41.2.3 “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:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read register 'f'</td>
<td>Process Data</td>
<td>Write to destination</td>
</tr>
</tbody>
</table>

**If skip:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

**If skip and followed by 2-word instruction:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

**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 ≠ 0;
  - PC = Address (HERE + 2)
DCFSNZ  Decrement f, skip if not 0
Syntax:    DCFSNZ  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 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 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 Section
41.2.3 “Byte-Oriented and Bit-
Oriented Instructions in Indexed Lit-
eral Offset Mode” for details.

Words: 1
Cycles: 1(2)
Note: 3 cycles if skip and followed
by a 2-word instruction.

Q Cycle Activity:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read register ‘f’</td>
<td>Process Data</td>
<td>Write to destination</td>
</tr>
</tbody>
</table>

If skip:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

If skip and followed by 2-word instruction:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

Example: HEREDCFSNZTEMP, 1, 0
ZERO:
NZERO:

Before Instruction
TEMP = ?

After Instruction
TEMP = TEMP – 1,
PC = Address (ZERO)
PC = Address (NZERO)

GOTO Unconditional Branch
Syntax:    GOTO   k
Operands:  0 ≤ k ≤ 1048575
Operation: k → PC<20:1>
Status Affected: None
Encoding:  1st word (k<7:0>)
           2nd word (k<19:8>)
1110 1111 k1kkk kkkk kkkk0
1111 k1kkk kkkk kkkkG
Description: GOTO allows an unconditional branch
anywhere within entire
2-Mbyte memory range. The 20-bit
value ‘k’ is loaded into PC<20:1>.
GOTO is always a 2-cycle
instruction.

Words: 2
Cycles: 2

Q Cycle Activity:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read literal ‘k’&lt;7:0&gt;,</td>
<td>No operation</td>
<td>Read literal ‘k’&lt;19:8&gt;, Write to PC</td>
</tr>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

Example: GOTO THERE
After Instruction
PC = Address (THERE)
### INCF Increment f

**Syntax:**

\[ \text{INCF } f \{,d \{,a\}\} \]

**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:**

\[
\begin{array}{c}
0010 \ 10da \ ffff \ ffff \\
\end{array}
\]

**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 ‘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 41.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.

**Words:**

1

**Cycles:**

1

**Q Cycle Activity:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read register ‘f’</td>
<td>Process Data</td>
<td>Write to destination</td>
</tr>
</tbody>
</table>

**Example:**

\[ \text{INCF CNT, 1, 0} \]

Before Instruction

\[
\begin{array}{l}
\text{CNT} = \text{FFh} \\
\text{Z} = 0 \\
\text{C} = ? \\
\text{DC} = ? \\
\end{array}
\]

After Instruction

\[
\begin{array}{l}
\text{CNT} = \text{00h} \\
\text{Z} = 1 \\
\text{C} = 1 \\
\text{DC} = 1 \\
\end{array}
\]

### INCFSZ Increment f, skip if 0

**Syntax:**

\[ \text{INCFSZ } f \{,d \{,a\}\} \]

**Operands:**

\[ 0 \leq f \leq 255 \]
\[ d \in [0,1] \]
\[ a \in [0,1] \]

**Operation:**

\[ (f) + 1 \rightarrow \text{dest}, \text{ skip if result } = 0 \]

**Status Affected:** None

**Encoding:**

\[
\begin{array}{c}
0011 \ 11da \ ffff \ ffff \\
\end{array}
\]

**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 ‘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 41.2.3 “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:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read register ‘f’</td>
<td>Process Data</td>
<td>Write to destination</td>
</tr>
</tbody>
</table>

If skip:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

If skip and followed by 2-word instruction:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

**Example:**

\[ \text{HERE \ INCFSZ CNT, 1, 0} \]
\[ \text{NZERO :} \]
\[ \text{ZERO :} \]

Before Instruction

\[
\begin{array}{l}
\text{PC} = \text{Address (HERE)} \\
\end{array}
\]

After Instruction

\[
\begin{array}{l}
\text{CNT} = \text{CNT + 1} \\
\text{PC} = \text{Address (ZERO)} \\
\text{If CNT} = \text{0;} \\
\text{PC} = \text{Address (NZERO)} \\
\end{array}
\]
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 Section 41.2.3 “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:

Example: HERE INFSNZ REG, 1, 0

ZERO NZERO

Before Instruction
PC = Address (HERE)

After Instruction
REG = REG + 1
If REG ≠ 0:
PC = Address (NZERO)
If REG = 0:
PC = Address (ZERO)

IORLW Inclusive OR literal with W

Syntax: IORLW k

Operands: 0 ≤ k ≤ 255

Operation: (W) .OR. k → W

Status Affected: N, Z

Encoding: 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:

Example: IORLW 35h

Before Instruction
W = 9Ah

After Instruction
W = BFh
IORWF  Inclusive OR W with f

Syntax: IORWF f, d {, a}

Operands: 0 ≤ f ≤ 255
          d ∈ [0, 1]
          a ∈ [0, 1]

Operation: (W) .OR. (f) → dest

Status Affected: N, Z

Encoding:

```
0001 00da ffff ffff
```

Description: 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).

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 Section 41.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.

Words: 1
Cycles: 1

Q Cycle Activity:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read</td>
<td>Process</td>
<td>Write</td>
</tr>
<tr>
<td>register 'f'</td>
<td>literal 'k'</td>
<td>Data</td>
<td>literal 'k'</td>
</tr>
</tbody>
</table>

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 ≤ f ≤ 2
          0 ≤ k ≤ 16383

Operation: k → FSRf

Status Affected: None

Encoding:

```
1110 1111 1110 0000 00k13k kkkk
```

Description: The 14-bit literal 'k' is loaded into the File Select Register pointed to by 'f'.

Words: 2
Cycles: 2

Example: LFSR 2, 3ABh

After Instruction

FSR2H = 03h
FSR2L = ABh
MOVF  Move f
Syntax: MOVF f{,d {,a}}
Operands: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]
Operation: f → 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 ≤ 95 (5Fh). See Section 41.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.

Example:
Before Instruction
REG = 22h
W = FFh
After Instruction
REG = 22h
W = 22h

Words: 1
Cycles: 1
Q Cycle Activity:

Example: MOVFF REG1, REG2
Before Instruction
REG1 = 33h
REG2 = 11h
After Instruction
REG1 = 33h
REG2 = 33h

MOVFF  Move f to f
Syntax: MOVFF f_s f_d
Operands: 0 ≤ f_s ≤ 4095
0 ≤ f_d ≤ 4095
Operation: (f_s) → f_d
Status Affected: None
Encoding:
1st word (source) 1100 1111 ffff ffff ffff ffff
2nd word (destin.) ffff ffff ffff ffff
Description: The contents of source register ‘f_s’ are moved to destination register ‘f_d’. Location of source ‘f_s’ can be anywhere in the 4096-byte data space (000h to FFFh) and location of destination ‘f_d’ 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:
MOVFFL  Move f to f (Long Range)

Syntax: MOVFFL  f_s, f_d

Operands: 0 ≤ f_s ≤ 16383
0 ≤ f_d ≤ 16383

Operation: (f_s) → f_d

Status Affected: None

Encoding:

<table>
<thead>
<tr>
<th>1st word</th>
<th>2nd word</th>
<th>3rd word</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>1111</td>
<td>1111</td>
</tr>
<tr>
<td></td>
<td>f_s, f_s, f_s</td>
<td>f_s, f_s, f_s</td>
</tr>
<tr>
<td></td>
<td>f_s, f_s, f_s</td>
<td>f_s, f_s, f_s</td>
</tr>
</tbody>
</table>

Description: The contents of source register 'f_s' are moved to destination register 'f_d'. Location of source 'f_s' 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.

Words: 3
Cycles: 3

Q Cycle Activity:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
<tr>
<td>Read register 'f_s' (src)</td>
<td>Process data</td>
<td>No operation</td>
<td></td>
</tr>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>Write register 'f_d' (dest)</td>
<td></td>
</tr>
</tbody>
</table>

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: MOVLB  k

Operands: 0 ≤ k ≤ 63

Operation: k → BSR

Status Affected: None

Encoding:

| 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>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read literal 'k'</td>
<td>Process Data</td>
<td>Write literal 'k' to BSR</td>
</tr>
</tbody>
</table>

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: \[0000\ 1110\ 0000\ 0000\]
Description: The 8-bit literal 'k' is loaded into W.

Words: 1
Cycles: 1
Q Cycle Activity:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read literal 'k'</td>
<td>Process Data</td>
<td>Write to W</td>
</tr>
</tbody>
</table>

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: \[0110\ 111a\ 0000\ 0000\]
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 41.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.

Words: 1
Cycles: 1
Q Cycle Activity:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Process Data</td>
<td>Write register 'f'</td>
<td></td>
</tr>
</tbody>
</table>

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) x k → PRODH:PRODL

**Status Affected:**
None

**Encoding:**
```
0000 1101 kkkk kkkk
```

**Description:**
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. 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.

**Words:**
1

**Cycles:**
1

**Q Cycle Activity:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read literal 'k'</td>
<td>Process Data</td>
<td>Write registers PRODH: PRODL</td>
</tr>
</tbody>
</table>

**Example:**
MULLW 0C4h

Before Instruction

<table>
<thead>
<tr>
<th>W</th>
<th>PRODH</th>
<th>PRODL</th>
</tr>
</thead>
<tbody>
<tr>
<td>E2h</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

After Instruction

<table>
<thead>
<tr>
<th>W</th>
<th>PRODH</th>
<th>PRODL</th>
</tr>
</thead>
<tbody>
<tr>
<td>E2h</td>
<td>ADh</td>
<td>08h</td>
</tr>
</tbody>
</table>

### MULWF
Multiply W with f

**Syntax:**
MULWF f{,a}

**Operands:**
0 ≤ f ≤ 255

**Operands:**
a ∈ [0,1]

**Operation:**
(W) x (f) → PRODH:PRODL

**Status Affected:**
None

**Encoding:**
```
0000 001a ffff ffff
```

**Description:**
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. 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.

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 Section 41.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.

**Words:**
1

**Cycles:**
1

**Q Cycle Activity:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read register 'f'</td>
<td>Process Data</td>
<td>Write registers PRODH: PRODL</td>
</tr>
</tbody>
</table>

**Example:**
MULWF REG, 1

Before Instruction

<table>
<thead>
<tr>
<th>W</th>
<th>REG</th>
<th>PRODH</th>
<th>PRODL</th>
</tr>
</thead>
<tbody>
<tr>
<td>C4h</td>
<td>B5h</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

After Instruction

<table>
<thead>
<tr>
<th>W</th>
<th>REG</th>
<th>PRODH</th>
<th>PRODL</th>
</tr>
</thead>
<tbody>
<tr>
<td>C4h</td>
<td>B5h</td>
<td>8Ah</td>
<td>94h</td>
</tr>
</tbody>
</table>
NEGF Negate f

Syntax:          NEGF     f {,a}
Operands:       0 ≤ f ≤ 255
                a ∈ {0, 1}
Operation:      (f) + 1 → f
Status Affected: N, OV, C, DC, Z
Encoding:       0110 110a ffff ffff
Description:   Location ‘f’ is negated using two’s complement. The result is placed in the
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. 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 Section 41.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.

Words:        1
Cycles:       1
Q Cycle Activity:

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:        0000 0000 0000 0000

Description:   No operation.

Words:        1
Cycles:       1
Q Cycle Activity:

Example:       None.
## POP
### Pop Top of Return Stack

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>POP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands:</td>
<td>None</td>
</tr>
<tr>
<td>Operation:</td>
<td>(TOS) → bit bucket</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>None</td>
</tr>
<tr>
<td>Encoding:</td>
<td>0000 0000 0000 0110</td>
</tr>
<tr>
<td>Description:</td>
<td>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. This instruction is provided to enable the user to properly manage the return stack to incorporate a software stack.</td>
</tr>
</tbody>
</table>

| Words: | 1 |
| Cycles: | 1 |

### Q Cycle Activity:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>No operation</td>
<td>POP TOS value</td>
<td>No operation</td>
</tr>
</tbody>
</table>

### Example:

**Before Instruction**

- TOS = 0031A2h
- Stack (1 level down) = 014332h

**After Instruction**

- TOS = 014332h
- PC = NEW

## PUSH
### Push Top of Return Stack

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>PUSH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands:</td>
<td>None</td>
</tr>
<tr>
<td>Operation:</td>
<td>(PC + 2) → TOS</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>None</td>
</tr>
<tr>
<td>Encoding:</td>
<td>0000 0000 0000 0101</td>
</tr>
<tr>
<td>Description:</td>
<td>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.</td>
</tr>
</tbody>
</table>

| Words: | 1 |
| Cycles: | 1 |

### Q Cycle Activity:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>PUSH PC + 2 onto return stack</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

### Example:

**Before Instruction**

- TOS = 345Ah
- PC = 0124h

**After Instruction**

- PC = 0126h
- TOS = 0126h
- Stack (1 level down) = 345Ah
### RCALL Relative Call

**Syntax:**
```plaintext
RCALL n
```

**Operands:**

-1024 ≤ n ≤ 1023

**Operation:**

\[(PC) + 2 \rightarrow TOS,
(\text{PC}) + 2 + 2n \rightarrow \text{PC}\]

**Status Affected:** None

**Encoding:**

```
1101 lnnn 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:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read literal ('n')</td>
<td>Process Data</td>
<td>Write to PC</td>
</tr>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

**Example:**

Before Instruction  
\(PC = \text{Address (HERE)}\)

After Instruction  
\(PC = \text{Address (Jump)}\)
\(TOS = \text{Address (HERE + 2)}\)

---

### RESET Reset

**Syntax:**

```
RESET
```

**Operands:** None

**Operation:** Reset all registers and flags that are affected by a MCLR Reset.

**Status Affected:** All

**Encoding:**

```
0000 0000 1111 1111
```

**Description:** This instruction provides a way to execute a MCLR Reset by software.

**Words:** 1

**Cycles:** 1

**Q Cycle Activity:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Start</td>
<td>No operation</td>
<td>No operation</td>
</tr>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

**Example:**

```
RESET
```

After Instruction  
\(\text{Registers = Reset Value}\)
\(\text{Flags* = Reset Value}\)
### RETFIE
**Return from Interrupt**

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>RETFIE {s}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands:</td>
<td>s ∈ [0, 1]</td>
</tr>
<tr>
<td>Operation:</td>
<td>(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.</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>STAT&lt;1:0&gt; in INTCON1 register</td>
</tr>
<tr>
<td>Encoding:</td>
<td>0000 0000 0001 000s</td>
</tr>
<tr>
<td>Description:</td>
<td>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).</td>
</tr>
</tbody>
</table>

**Words:** 1  
**Cycles:** 2  
**Q Cycle Activity:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

**Example:**

```assembly
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 = PRODH_SHAD
- PCLATH/U = PCLATH/U_SHAD

### RETLW
**Return literal to W**

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>RETLW k</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands:</td>
<td>0 ≤ k ≤ 255</td>
</tr>
<tr>
<td>Operation:</td>
<td>k → W, (TOS) → PC, PCLATU, PCLATH are unchanged</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>None</td>
</tr>
<tr>
<td>Encoding:</td>
<td>0000 1100 kkkk kkkk</td>
</tr>
<tr>
<td>Description:</td>
<td>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.</td>
</tr>
</tbody>
</table>

**Words:** 1  
**Cycles:** 2  
**Q Cycle Activity:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

**Example:**

```assembly
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: \[
\text{RETURN \{s\}}
\]

Operands: \[s \in [0,1]\]

Operation:

- \[(\text{TOS}) \rightarrow \text{PC},\]
- \[(\text{WS}) \rightarrow \text{W},\]
- \[(\text{STATUSS}) \rightarrow \text{Status},\]
- \[(\text{BSRS}) \rightarrow \text{BSR},\]
- \[\text{PCLATU, PCLATH are unchanged}\]

Status Affected: None

Encoding:

| 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, STATUSS 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:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>No operation</td>
<td>Process Data</td>
<td>POP PC from stack</td>
</tr>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

Example: \[
\text{RETURN} \]

After Instruction:

\[\text{PC} = \text{TOS}\]

---

**RLCF**

Rotate Left f through Carry

Syntax: \[
\text{RLCF \{d, (a)\}}
\]

Operands:

- \[0 \leq f \leq 255\]
- \[d \in [0,1]\]
- \[a \in [0,1]\]

Operation:

- \[(f<<n) \rightarrow \text{dest}<n + 1>,\]
- \[(f<<7) \rightarrow \text{C},\]
- \[(\text{C}) \rightarrow \text{dest}<0>\]

Status Affected: C, N, Z

Encoding:

| 0011 | 01da | ffff | ffff |

Description:

The contents of register 'f' are rotated one bit to the left through the CARRY flag. If \[d = 0\], the result is placed in W. If \[d = 1\], the result is stored back in register 'f' (default). If \[a = 0\], the Access Bank is selected. If \[a = 1\], the BSR is used to select the GPR bank.

If \[a = 0\] and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever \[f \leq 95\] (5Fh). See Section 41.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.

Words: 1

Cycles: 1

Q Cycle Activity:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read register f</td>
<td>Process Data</td>
<td>Write to destination</td>
</tr>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

Example: \[
\text{RLCF \ REG, 0, 0} \]

Before Instruction

\[
\text{REG} = 1110 0110 \\
\text{C} = 0
\]

After Instruction

\[
\text{REG} = 1110 0110 \\
\text{W} = 1100 1100 \\
\text{C} = 1
\]
### RLNCF Rotate Left f (No Carry)

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>RLNCF  \ f,!d,(!a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands:</td>
<td>(0 \leq f \leq 255)</td>
</tr>
<tr>
<td>(d \in [0,1])</td>
<td></td>
</tr>
<tr>
<td>(a \in [0,1])</td>
<td></td>
</tr>
<tr>
<td>Operation:</td>
<td>((f_{n}) \rightarrow \text{dest}_{n+1},)</td>
</tr>
<tr>
<td>((f_{7}) \rightarrow \text{dest}_{0})</td>
<td></td>
</tr>
<tr>
<td>Status Affected:</td>
<td>N, Z</td>
</tr>
<tr>
<td>Encoding:</td>
<td></td>
</tr>
<tr>
<td>Description:</td>
<td>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. 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 41.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Example:</th>
<th>RLNCF  REG, 1, 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before Instruction</td>
<td></td>
</tr>
<tr>
<td>REG</td>
<td>1010 1011</td>
</tr>
<tr>
<td>After Instruction</td>
<td></td>
</tr>
<tr>
<td>REG</td>
<td>0101 0111</td>
</tr>
</tbody>
</table>

### RRCF Rotate Right f through Carry

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>RRCF  \ f,!d,(!a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands:</td>
<td>(0 \leq f \leq 255)</td>
</tr>
<tr>
<td>(d \in [0,1])</td>
<td></td>
</tr>
<tr>
<td>(a \in [0,1])</td>
<td></td>
</tr>
<tr>
<td>Operation:</td>
<td>((f_{n}) \rightarrow \text{dest}_{n-1},)</td>
</tr>
<tr>
<td>((f_{0}) \rightarrow C,)</td>
<td></td>
</tr>
<tr>
<td>((C) \rightarrow \text{dest}_{7})</td>
<td></td>
</tr>
<tr>
<td>Status Affected:</td>
<td>C, N, Z</td>
</tr>
<tr>
<td>Encoding:</td>
<td></td>
</tr>
<tr>
<td>Description:</td>
<td>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). 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 41.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Example:</th>
<th>RRCF  REG, 0, 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before Instruction</td>
<td></td>
</tr>
<tr>
<td>REG</td>
<td>1110 0110</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
</tr>
<tr>
<td>After Instruction</td>
<td></td>
</tr>
<tr>
<td>REG</td>
<td>1110 0110</td>
</tr>
<tr>
<td>W</td>
<td>0111 0011</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
</tr>
</tbody>
</table>
**RRNCF**  
**Rotate Right f (No Carry)**  

**Syntax:**  
RRNCF  \( f \{,d \{,a\}\} \)  

**Operands:**  
- \( 0 \leq f \leq 255 \)  
- \( d \in [0,1] \)  
- \( a \in [0,1] \)  

**Operation:**  
- \( (f_{<n>} \rightarrow \text{dest}_{<n-1>}, \)  
- \( f_{<0>} \rightarrow \text{dest}_{<7>} \)  

**Status Affected:**  
N, Z  

**Encoding:**  
| 0100 | 00da | ffff | ffff |  

**Description:**  
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 \( f \leq 95 \) (5Fh). See Section 41.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.

**Words:**  
1  

**Cycles:**  
1  

**Q Cycle Activity:**  

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read register 'f'</td>
<td>Process Data</td>
<td>Write to destination</td>
</tr>
</tbody>
</table>

**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:**  
- \( 0 \leq f \leq 255 \)  
- \( a \in [0,1] \)  

**Operation:**  
FFH \( \rightarrow \) f  

**Status Affected:**  
None  

**Encoding:**  
| 0110 | 100a | ffff | ffff |  

**Description:**  
The contents of the specified register are set to FFh.  
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 41.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.

**Words:**  
1  

**Cycles:**  
1  

**Q Cycle Activity:**  

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read register 'f'</td>
<td>Process Data</td>
<td>Write register 'f'</td>
</tr>
</tbody>
</table>

**Example:**  
SETF  REG, 1  

Before Instruction  
REG = 5Ah  
After Instruction  
REG = FFH
### SLEEP
Enter Sleep mode

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>SLEEP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands:</td>
<td>None</td>
</tr>
<tr>
<td>Operation:</td>
<td>00h → WDT,</td>
</tr>
<tr>
<td></td>
<td>0 → WDT postscaler,</td>
</tr>
<tr>
<td></td>
<td>1 → TO,</td>
</tr>
<tr>
<td></td>
<td>0 → PD</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>TO, PD</td>
</tr>
<tr>
<td>Encoding:</td>
<td>0000 0000 0000 0011</td>
</tr>
<tr>
<td>Description:</td>
<td>The Power-down Status bit (PD) is cleared. The Time-out Status bit (TO) is set. Watchdog Timer and its postscaler are cleared. The processor is put into Sleep mode with the oscillator stopped.</td>
</tr>
<tr>
<td>Words:</td>
<td>1</td>
</tr>
<tr>
<td>Cycles:</td>
<td>1</td>
</tr>
<tr>
<td>Q Cycle Activity:</td>
<td></td>
</tr>
</tbody>
</table>

#### Example:

**Before Instruction**

<table>
<thead>
<tr>
<th>Register</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TO</td>
<td>?</td>
</tr>
<tr>
<td>PD</td>
<td>?</td>
</tr>
</tbody>
</table>

**After Instruction**

<table>
<thead>
<tr>
<th>Register</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TO</td>
<td>1†</td>
</tr>
<tr>
<td>PD</td>
<td>0</td>
</tr>
</tbody>
</table>

† If WDT causes wake-up, this bit is cleared.

### SUBFSR
Subtract Literal from FSR

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>SUBFSR f, k</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands:</td>
<td>0 ≤ k ≤ 63</td>
</tr>
<tr>
<td></td>
<td>f ∈ {0, 1, 2}</td>
</tr>
<tr>
<td>Operation:</td>
<td>FSR(f) – k → FSRf</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>None</td>
</tr>
<tr>
<td>Encoding:</td>
<td>[1110 1001 \text{ffkk kkkk}]</td>
</tr>
<tr>
<td>Description:</td>
<td>The 6-bit literal 'k' is subtracted from the contents of the FSR specified by 'f'.</td>
</tr>
<tr>
<td>Words:</td>
<td>1</td>
</tr>
<tr>
<td>Cycles:</td>
<td>1</td>
</tr>
<tr>
<td>Q Cycle Activity:</td>
<td></td>
</tr>
</tbody>
</table>

#### Example:

**Before Instruction**

<table>
<thead>
<tr>
<th>Register</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSR2</td>
<td>03FFh</td>
</tr>
</tbody>
</table>

**After Instruction**

<table>
<thead>
<tr>
<th>Register</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSR2</td>
<td>03DCb</td>
</tr>
</tbody>
</table>
SUBFWB

Subtract f from W with borrow

Syntax: SUBFWB f {,d {,a}}

Operands:
0 ≤ f ≤ 255
 d ∈ [0,1]
 a ∈ [0,1]

Operation: (W) – (f) – (C) → dest

Status Affected: N, OV, C, DC, Z

Encoding:
0000 0101 10da ffff fffe

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 ≤ 95 (5Fh). See Section 41.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.

Words: 1

Cycles: 1

Q Cycle Activity:

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
N = 0 ; result is zero

SUBLW

Subtract W from literal

Syntax: SUBLW k

Operands: 0 ≤ k ≤ 255

Operation: k – (W) → W

Status Affected: N, OV, C, DC, Z

Encoding:
0000 1000 kkkh kkkh

Description:
W is subtracted from the 8-bit literal ‘k’. The result is placed in W.

Words: 1

Cycles: 1

Q Cycle Activity:

Example 1:
SUBLW 02h

Before Instruction
W = 01h
C = ?

After Instruction
W = 00h
C = 1 ; result is zero
Z = 1
N = 0

Example 2:
SUBLW 02h

Before Instruction
W = 02h
C = ?

After Instruction
W = FFh ; (2’s complement)
C = 0 ; result is negative
Z = 0
N = 1

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 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

Operation: (f) – (W) → dest

Status Affected: N, OV, C, DC, Z

Encoding: 0101 11da ffff ffff

Description: 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).

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 Section 41.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.

Words: 1
Cycles: 1

Q Cycle Activity:

Example 1: SUBWF REG, 1, 0
Before Instruction
REG = 3
W = 2
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 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

Operation: (f) – (W) – (C) → dest

Status Affected: N, OV, C, DC, Z

Encoding: 0101 10da ffff ffff

Description: 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.

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 Section 41.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.

Words: 1
Cycles: 1

Q Cycle Activity:

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 = 1 ; 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)
W = 0Eh (0000 1110)
C = 0
Z = 0
N = 1 ; result is negative
SWAPF Swap f

Syntax: SWAPF f {,d {,a}}

Operands:
- 0 ≤ f ≤ 255
- d ∈ [0,1]
- a ∈ [0,1]

Operation:
- (f<3:0>) → dest<7:4>
- (f<7:4>) → 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 ≤ 95 (5Fh). See Section 41.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.

Words: 1
Cycles: 1

Q Cycle Activity:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read register 'f'</td>
<td>Process Data</td>
<td>Write to destination</td>
</tr>
</tbody>
</table>

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 +*, (Prog Mem (TBLPTR)) → TABLAT;
  - (TBLPTR) + 1 → TBLPTR;

**Status Affected:** None

**Encoding:**

<table>
<thead>
<tr>
<th>0000</th>
<th>0000</th>
<th>0000</th>
<th>10nn</th>
</tr>
</thead>
<tbody>
<tr>
<td>nn=0</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td>+</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-2</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-3</td>
<td>+</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
<tr>
<td>No operation</td>
<td>No operation (Read Program Memory)</td>
<td>No operation</td>
<td>No operation (Write TABLAT)</td>
</tr>
</tbody>
</table>

---

**Example1:**

**Before Instruction**

- TABLAT = 55h
- MEMORY (00A356h) = 34h

**After Instruction**

- TABLAT = 34h
- TBLPTR = 00A357h

**Example2:**

**Before Instruction**

- TABLAT = AAh
- TBLPTR = 01A357h
- MEMORY (01A358h) = 34h

**After Instruction**

- TABLAT = 34h
- TBLPTR = 01A358h
TBLWT Table Write

Syntax: TBLWT (*; *+; *-; +*)

Operands: None

Operation:
- if TBLWT*, (TABLAT) → Holding Register; TBLPTR – No Change;
- if TBLWT*+, (TABLAT) → Holding Register; (TBLPTR) + 1 → TBLPTR;
- if TBLWT*-,(TABLAT) → Holding Register; (TBLPTR) – 1 → TBLPTR;
- if TBLWT+*, (TBLPTR) + 1 → TBLPTR;

Status Affected: None

Encoding:

<table>
<thead>
<tr>
<th>0000</th>
<th>0000</th>
<th>0000</th>
<th>linn</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>nn=0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-3</td>
</tr>
</tbody>
</table>

Description: 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 Section 13.1 "Program Flash Memory" for additional details on programming Flash memory.)

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.

TBLPTR[0] = 0: Least Significant Byte of Program Memory Word
TBLPTR[0] = 1: Most Significant Byte of Program Memory Word

The TBLWT instruction can modify the value of TBLPTR as follows:
- no change
- post-increment
- post-decrement
- pre-increment

Example 1:

Before Instruction

<table>
<thead>
<tr>
<th>TABLET</th>
<th>TBLPTR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>00A356h</td>
</tr>
</tbody>
</table>

After Instructions (table write completion)

<table>
<thead>
<tr>
<th>TABLET</th>
<th>TBLPTR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>00A356h</td>
</tr>
<tr>
<td></td>
<td>55h</td>
</tr>
</tbody>
</table>

Example 2:

Before Instruction

<table>
<thead>
<tr>
<th>TABLET</th>
<th>TBLPTR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>00A357h</td>
</tr>
</tbody>
</table>

After Instructions (table write completion)

<table>
<thead>
<tr>
<th>TABLET</th>
<th>TBLPTR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>00A357h</td>
</tr>
<tr>
<td>01389Ah</td>
<td>34h</td>
</tr>
</tbody>
</table>

Words: 1
Cycles: 2

Q Cycle Activity:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
<tr>
<td>No operation</td>
<td>No operation (Read TABLAT)</td>
<td>No operation</td>
<td>No operation (Write to Holding Register)</td>
</tr>
</tbody>
</table>
TSTFSZ  Test f, skip if 0
Syntax: TSTFSZ f, {a}
Operands: 0 ≤ f ≤ 255
           a ∈ [0, 1]
Operation: skip if f = 0
Status Affected: None
Encoding: 0110 011a ffff ffff
Description: 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.
              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 Section
              41.2.3 “Byte-Oriented and Bit-
              Oriented Instructions in Indexed Lit-
              eral Offset Mode” for details.
Words: 1
Cycles: 1(2)
Note: 3 cycles if skip and followed
       by a 2-word instruction.
Q Cycle Activity:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read</td>
<td>Process</td>
<td>No operation</td>
</tr>
<tr>
<td>register 'f'</td>
<td>Data</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If skip:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

If skip and followed by 2-word instruction:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

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 ≤ k ≤ 255
Operation: (W) .XOR. k → W
Status Affected: N, Z
Encoding: 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:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read</td>
<td>Process</td>
<td>Data</td>
</tr>
<tr>
<td>literal ‘k’</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example: XORLW 0AFh

Before Instruction
W = B5h

After Instruction
W = 1Ah
**XORWF**  
**Exclusive OR W with f**

**Syntax:**  
`XORWF f {,d {,a}}`

**Operands:**  
- `0 \leq f \leq 255`
- `d \in [0,1]`
- `a \in [0,1]`

**Operation:**  
`(W) \cdot XOR. (f) \rightarrow dest`

**Status Affected:**  
`N, Z`

**Encoding:**  
```
0001 10da ffff ffff
```

**Description:**  
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). 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 41.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.

**Words:**  
1

**Cycles:**  
1

**Q Cycle Activity:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read</td>
<td>Process</td>
<td>Write to destination</td>
</tr>
<tr>
<td>register <code>f</code></td>
<td>Data</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Example:**  
`XORWF REG, 1, 0`

**Before Instruction**

- `REG = AFh`
- `W = B5h`

**After Instruction**

- `REG = 1Ah`
- `W = B5h`
41.2 Extended Instruction Set

In addition to the standard 75 instructions of the PIC18 instruction set, PIC18(L)F2x/4x/5xK42 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 (ADDFULNK 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 41-3. Detailed descriptions are provided in Section 41.2.2 “Extended Instruction Set”. The opcode field descriptions in Table 41-1 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.

41.2.1 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 41.2.3.1 “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 41-3: EXTENSIONS TO THE PIC18 INSTRUCTION SET

<table>
<thead>
<tr>
<th>Mnemonic, Operands</th>
<th>Description</th>
<th>Cycles</th>
<th>16-Bit Instruction Word</th>
<th>Status Affected</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>MSb</td>
<td>LSb</td>
</tr>
<tr>
<td>ADDULNK k</td>
<td>Add FSR2 with (k) &amp; return</td>
<td>2</td>
<td>1110 1000 11kk kkkk</td>
<td>None</td>
</tr>
<tr>
<td>MOVSF z, fd</td>
<td>Move z, (source) to 1st word</td>
<td>2</td>
<td>1110 1011 0zzz zzzz</td>
<td>None</td>
</tr>
<tr>
<td>MOVSFL z, fd</td>
<td>Opcode 1st word</td>
<td>2</td>
<td>1111 ffff ffff ffff</td>
<td>None</td>
</tr>
<tr>
<td>MOVSS z, zd</td>
<td>Move z (source) to 1st word</td>
<td>3</td>
<td>0000 0000 0000 0010</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>f (destination) 2nd word</td>
<td>2</td>
<td>1111 xxxx zzzz zzzf</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>f (full destination) 3rd word</td>
<td>1</td>
<td>1111 ffff ffff ffff</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>z (destination) 2nd word</td>
<td>2</td>
<td>1111 xxxx xxxx zzzz</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>f (destination) 3rd word</td>
<td>1</td>
<td>1110 1011 1zzz zzzz</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>f (full destination) 3rd word</td>
<td>1</td>
<td>1110 1010 kkkk kkkk</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>z (destination) 2nd word</td>
<td>2</td>
<td>1110 1001 11kk kkkk</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>k (return)</td>
<td>2</td>
<td>1110 1001 11kk kkkk</td>
<td>None</td>
</tr>
</tbody>
</table>

**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**.

**Note 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.

**Note 3:** Only available when extended instruction set is enabled.

**Note 4:** f and f 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.
41.2.2 EXTENDED INSTRUCTION SET

<table>
<thead>
<tr>
<th>ADDULNK</th>
<th>Add Literal to FSR2 and Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>ADDULNK k</td>
</tr>
<tr>
<td>Operands:</td>
<td>0 ≤ k ≤ 63</td>
</tr>
<tr>
<td>Operation:</td>
<td>FSR2 + k → FSR2.</td>
</tr>
<tr>
<td></td>
<td>(TOS) → PC</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>None</td>
</tr>
<tr>
<td>Encoding:</td>
<td>1110 1000 11kk kkkk</td>
</tr>
<tr>
<td>Description:</td>
<td>The 6-bit literal 'k' is added to the contents of 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 ADDFSR instruction, where f = 3 (binary '11'); it operates only on FSR2.</td>
</tr>
<tr>
<td>Words:</td>
<td>1</td>
</tr>
<tr>
<td>Cycles:</td>
<td>2</td>
</tr>
</tbody>
</table>

**Q Cycle Activity:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Decode</td>
<td>Read literal 'k'</td>
<td>Process Data</td>
<td>Write to FSR</td>
</tr>
</tbody>
</table>

**Example:**

ADDULNK 23h

<table>
<thead>
<tr>
<th>Before Instruction</th>
<th>After Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSR2 = 03FFh</td>
<td>FSR2 = 0422h</td>
</tr>
<tr>
<td>PC = 0100h</td>
<td>PC = (TOS)</td>
</tr>
</tbody>
</table>

**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 [zₙ], fᵟ

**Operands:**

0 ≤ zₙ ≤ 127
0 ≤ fᵟ ≤ 4095

**Operation:**

((FSR2) + zₙ) → fᵟ

**Status Affected:**

None

**Encoding:**

<table>
<thead>
<tr>
<th>1st word (source)</th>
<th>2nd word (destin.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1110 1111 0zzz zzzz</td>
<td>ffff ffff ffff ffff</td>
</tr>
</tbody>
</table>

**Description:**

The contents of the source register are moved to destination register 'fᵟ'. The actual address of the source register is determined by adding the 7-bit literal offset 'zₙ' in the first word to the value of FSR2. The address of the destination register is specified by the 12-bit literal 'fᵟ' in the second word. Both addresses can be anywhere in the 4096-byte data space (000h to FFFh).

**Example:**

MOVSF [05h], REG2

Before Instruction

<table>
<thead>
<tr>
<th>FSR2</th>
<th>Contents of 85h</th>
<th>REG2</th>
</tr>
</thead>
<tbody>
<tr>
<td>80h</td>
<td>33h</td>
<td>11h</td>
</tr>
</tbody>
</table>

After Instruction

<table>
<thead>
<tr>
<th>FSR2</th>
<th>Contents of 85h</th>
<th>REG2</th>
</tr>
</thead>
<tbody>
<tr>
<td>80h</td>
<td>33h</td>
<td>33h</td>
</tr>
</tbody>
</table>

MOVSFL Move Indexed to f (Long Range)

**Syntax:**

MOVSFL [zₙ], fᵟ

**Operands:**

0 ≤ zₙ ≤ 127
0 ≤ fᵟ ≤ 16383

**Operation:**

((FSR2) + zₙ) → fᵟ

**Status Affected:**

None

**Encoding:**

<table>
<thead>
<tr>
<th>1st word (opcode)</th>
<th>2nd word (source)</th>
<th>3rd word (full destin.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>1111 xxxz zzzz zzsff</td>
<td>ffff ffff ffff ffff</td>
</tr>
</tbody>
</table>

**Description:**

The contents of the source register are moved to destination register 'fᵟ'. The actual address of the source register is determined by adding the 7-bit literal offset 'zₙ' in the first word to the value of FSR2 (14 bits). The address of the destination register is specified by the 14-bit literal 'fᵟ' 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.

**Example:**

MOVSFL [05h], REG2

Before Instruction

<table>
<thead>
<tr>
<th>FSR2</th>
<th>Contents of 85h</th>
<th>REG2</th>
</tr>
</thead>
<tbody>
<tr>
<td>80h</td>
<td>33h</td>
<td>11h</td>
</tr>
</tbody>
</table>

After Instruction

<table>
<thead>
<tr>
<th>FSR2</th>
<th>Contents of 85h</th>
<th>REG2</th>
</tr>
</thead>
<tbody>
<tr>
<td>80h</td>
<td>33h</td>
<td>33h</td>
</tr>
</tbody>
</table>
**MOVSS**

Move Indexed to Indexed

**Syntax:**
```
MOVSS [zs], [zd]
```

**Operands:**
- `0 ≤ zs ≤ 127`
- `0 ≤ zd ≤ 127`

**Operation:**
```
((FSR2) + zs) → ((FSR2) + zd)
```

**Status Affected:** None

**Encoding:**

<table>
<thead>
<tr>
<th>1st word (source)</th>
<th>2nd word (dest.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1110</td>
<td>1011</td>
</tr>
<tr>
<td>1111</td>
<td>xxxx</td>
</tr>
<tr>
<td>1zzz</td>
<td>xzzz</td>
</tr>
<tr>
<td>zzzzs</td>
<td>zzzzd</td>
</tr>
</tbody>
</table>

**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 `zs` or `zd`, 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:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Determine source addr</td>
<td>Determine source addr</td>
<td>Read source reg</td>
</tr>
<tr>
<td>Decode</td>
<td>Determine dest addr</td>
<td>Determine dest addr</td>
<td>Write to dest reg</td>
</tr>
</tbody>
</table>

**Example:**
```
MOVSS [05h], [06h]
```

**Before Instruction**

<table>
<thead>
<tr>
<th>FSR2</th>
<th>Contents of 85h</th>
<th>Contents of 86h</th>
</tr>
</thead>
<tbody>
<tr>
<td>80h</td>
<td>33h</td>
<td>11h</td>
</tr>
</tbody>
</table>

**After Instruction**

<table>
<thead>
<tr>
<th>FSR2</th>
<th>Contents of 85h</th>
<th>Contents of 86h</th>
</tr>
</thead>
<tbody>
<tr>
<td>80h</td>
<td>33h</td>
<td>33h</td>
</tr>
</tbody>
</table>

---

**PUSHL**

Store Literal at FSR2, Decrement FSR2

**Syntax:**
```
PUSHL k
```

**Operands:**
- `0 ≤ k ≤ 255`

**Operation:**
```
k → (FSR2),
FSR2 – 1 → FSR2
```

**Status Affected:** None

**Encoding:**

<table>
<thead>
<tr>
<th>1st word (source)</th>
<th>2nd word (dest.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111</td>
<td>1010</td>
</tr>
<tr>
<td>kkkk</td>
<td>kkkk</td>
</tr>
</tbody>
</table>

**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:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read 'k'</td>
<td>Process data</td>
<td>Write to destination</td>
</tr>
</tbody>
</table>

**Example:**
```
PUSHL 08h
```

**Before Instruction**

<table>
<thead>
<tr>
<th>FSR2H:FSR2L</th>
<th>Memory (01ECh)</th>
</tr>
</thead>
<tbody>
<tr>
<td>01ECh</td>
<td>00h</td>
</tr>
</tbody>
</table>

**After Instruction**

<table>
<thead>
<tr>
<th>FSR2H:FSR2L</th>
<th>Memory (01EBh)</th>
</tr>
</thead>
<tbody>
<tr>
<td>01EBh</td>
<td>08h</td>
</tr>
</tbody>
</table>
SUBULNK Subtracts Literal from FSR2 and Return

Syntax: SUBULNK k
Operands: \(0 \leq k \leq 63\)
Operation: FSR2 \(- k\) \(\rightarrow\) FSR2
(TOS) \(\rightarrow\) 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:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read register ('f')</td>
<td>Process Data</td>
<td>Write to destination</td>
</tr>
<tr>
<td>No Operation</td>
<td>No Operation</td>
<td>No Operation</td>
<td>No Operation</td>
</tr>
</tbody>
</table>

Example: SUBULNK 23h

Before Instruction
- FSR2 = 03FFh
- PC = 0100h

After Instruction
- FSR2 = 03DCh
- PC = (TOS)
41.2.3  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 41.2.3.1 “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.

41.2.3.1  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.

41.2.4  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)F2x/4x/5xK42, 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
(Indexed Literal Offset mode)

Syntax: ADDWF [k] {,d}
Operands: 0 ≤ k ≤ 95
         d ∈ [0,1]
Operation: (W) + ((FSR2) + k) → dest
Status Affected: N, OV, C, DC, Z
Encoding: 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’. 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:

Example:

Before Instruction
W = 17h
OFST = 2Ch
FSR2 = 0A00h
Contents of 0A2Ch = 20h
After Instruction
W = 37h
Contents of 0A2Ch = 20h

BSF

Bit Set Indexed
(Indexed Literal Offset mode)

Syntax: BSF [k], b
Operands: 0 ≤ f ≤ 95
         0 ≤ b ≤ 7
Operation: 1 → ((FSR2) + k)<b>
Status Affected: None
Encoding: 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:

Example:

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 ≤ k ≤ 95
Operation: FFh → ((FSR2) + k)
Status Affected: None
Encoding: 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:

Example:

Before Instruction
OFST = 2Ch
FSR2 = 0A00h
Contents of 0A2Ch = 00h
After Instruction
Contents of 0A2Ch = FFh
41.2.5 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)F2x/4x/5xK42 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.
### 42.0 REGISTER SUMMARY

| Table 42-1: REGISTER FILE SUMMARY FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES |
|---|---|---|---|---|---|---|---|---|---|
| Address | 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 | 37 |
| 3FFeh | TOSH | — | — | — | — | — | — | — | — | Top of Stack High byte | 37 |
| 3FFdh | TOSL | — | — | — | — | — | — | — | — | Top of Stack Low byte | 37 |
| 3FFCh | STKPTR | — | — | — | — | — | — | — | — | Stack Pointer | 39 |
| 3FFBh | PCLATH | — | — | — | — | — | — | — | — | Holding Register for PC High byte | 36 |
| 3FFAh | PCL | — | — | — | — | — | — | — | — | PC Low byte | 36 |
| 3FFFh | TBLPTRU | — | — | — | — | — | — | — | — | Program Memory Table Pointer Upper byte | 192 |
| 3FF7h | TBLPTRH | — | — | — | — | — | — | — | — | Program Memory Table Pointer High byte | 192 |
| 3FF6h | TBLPTRC | — | — | — | — | — | — | — | — | Program Memory Table Pointer Low byte | 192 |
| 3FF5h | TABLAT | — | — | — | — | — | — | — | — | Table Latch | 192 |
| 3FF4h | PRODH | — | — | — | — | — | — | — | — | Product Register High byte | 187 |
| 3FF3h | PRODL | — | — | — | — | — | — | — | — | Product Register Low byte | 187 |
| 3FF2h | — | — | — | — | — | — | — | — | — | Unimplemented | |
| 3FF1h | PCON1 | — | — | — | — | — | — | — | — | — | — | 91 |
| 3FF0h | PCON0 | STKOVF | STKUFN | WDTC | RWDT | RMCLR | Ri | POR | BOR | Working Register | 90 |
| 3FFEh | INDF0 | Uses contents of FSR0 to address data memory – value of FSR0 not changed | 60 |
| 3FECh | POSTINC0 | Uses contents of FSR0 to address data memory – value of FSR0 post-incremented | 61 |
| 3FEDh | POSTDEC0 | Uses contents of FSR0 to address data memory – value of FSR0 post-decremented | 61 |
| 3FEbh | PREINC0 | Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented | 61 |
| 3FEBh | PLUSW0 | Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented – value of FSR0 offset by W | 61 |
| 3FEa | FSR0H | — | — | — | — | — | — | — | — | Indirect Data Memory Address Pointer 0 High | 61 |
| 3FE9h | FSR0L | — | — | — | — | — | — | — | — | Indirect Data Memory Address Pointer 0 Low | 61 |
| 3FE8h | WREG | — | — | — | — | — | — | — | — | Working Register | 44 |
| 3FE7h | INDF1 | Uses contents of FSR1 to address data memory – value of FSR1 not changed | 61 |
| 3FE6h | POSTINC1 | Uses contents of FSR1 to address data memory – value of FSR1 post-incremented | 61 |
| 3FE5h | POSTDEC1 | Uses contents of FSR1 to address data memory – value of FSR1 post-decremented | 61 |
| 3FE4h | PREINC1 | Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented | 61 |
| 3FE3h | PLUSW1 | Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented – value of FSR1 offset by W | 61 |
| 3FE2h | FSR1H | — | — | — | — | — | — | — | — | Indirect Data Memory Address Pointer 1 High | 61 |
| 3FE1h | FSR1L | — | — | — | — | — | — | — | — | Indirect Data Memory Address Pointer 1 Low | 61 |
| 3FE0h | BSRR | — | — | — | — | — | — | — | — | Bank Select Register | 58 |
| 3FDFh | INDF2 | Uses contents of FSR2 to address data memory – value of FSR2 not changed | 61 |
| 3FDEh | POSTINC2 | Uses contents of FSR2 to address data memory – value of FSR2 post-incremented | 61 |
| 3FDDh | POSTDEC2 | Uses contents of FSR2 to address data memory – value of FSR2 post-decremented | 61 |
| 3FDCh | PREINC2 | Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented | 61 |
| 3FDBh | PLUSW2 | Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented – value of FSR2 offset by W | 61 |
| 3FDAh | FSR2H | — | — | — | — | — | — | — | — | Indirect Data Memory Address Pointer 2 High | 61 |
| 3FD9h | FSR2L | — | — | — | — | — | — | — | — | Indirect Data Memory Address Pointer 2 Low | 61 |
| 3FD8h | STATUS | — | — | — | — | — | — | — | — | Indirect Data Memory Address Pointer 2 Low | |
| 3FD7h | IVTBASEU | — | — | — | — | — | — | — | — | BASE20 | BASE19 | BASE18 | BASE17 | BASE16 | 166 |
| 3FD6h | IVTBASEH | BASE15 | BASE14 | BASE13 | BASE12 | BASE11 | BASE10 | BASE9 | BASE8 | 166 |
| 3FD5h | IVTBASEL | BASE7 | BASE6 | BASE5 | BASE4 | BASE3 | BASE2 | BASE1 | BASE0 | 166 |
| 3FD4h | IVTLOCK | — | — | — | — | — | — | — | — | — | — | IVTLOCKED | 168 |
| 3FD3h | INTCON1 | STAT | — | — | — | — | — | — | — | — | — | — | 136 |
| 3FD2h | INTCON0 | GIE | GIEL | IPEN | — | — | — | — | — | — | — | INT2EDG | INT1EDG | INT0EDG | 135 |

Legend:  
x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

Note:  
1: Unimplemented in LF devices.
2: Unimplemented in PIC18(L)F26/27K42.
4: Unimplemented in PIC18(L)F45/55K42.
### TABLE 42-1: REGISTER FILE SUMMARY FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES

<table>
<thead>
<tr>
<th>Address on page</th>
<th>Address</th>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register</th>
<th>Legend:</th>
<th>Note 1:</th>
<th>Note 2:</th>
<th>Note 3:</th>
<th>Note 4:</th>
</tr>
</thead>
<tbody>
<tr>
<td>3FDh - 3FD0h</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FCFh</td>
<td>PORTF(3)</td>
<td>RF7</td>
<td>RF6</td>
<td>RF5</td>
<td>RF4</td>
<td>RF3</td>
<td>RF2</td>
<td>RF1</td>
<td>RF0</td>
<td>—</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FCEh</td>
<td>PORTE</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>RE0</td>
<td>RE2(2)</td>
<td>RE1(2)</td>
<td>RE1(2)</td>
<td>—</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FCDh</td>
<td>PORTD(2)</td>
<td>RD7</td>
<td>RD6</td>
<td>RD5</td>
<td>RD4</td>
<td>RD3</td>
<td>RD2</td>
<td>RD1</td>
<td>RD0</td>
<td>—</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FCBh</td>
<td>PORTE</td>
<td>RC7</td>
<td>RC6</td>
<td>RC5</td>
<td>RC4</td>
<td>RC3</td>
<td>RC2</td>
<td>RC1</td>
<td>RC0</td>
<td>—</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FCAh</td>
<td>PORTA</td>
<td>RA7</td>
<td>RA6</td>
<td>RA5</td>
<td>RA4</td>
<td>RA3</td>
<td>RA2</td>
<td>RA1</td>
<td>RA0</td>
<td>—</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FChh - 3FC0h</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FB7h</td>
<td>TRISF(3)</td>
<td>TRISF7</td>
<td>TRISF6</td>
<td>TRISF5</td>
<td>TRISF4</td>
<td>TRISF3</td>
<td>TRISF2</td>
<td>TRISF1</td>
<td>TRISF0</td>
<td>3FBDh</td>
<td>264</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FB6h</td>
<td>TRISE(2)</td>
<td>TRIS7</td>
<td>TRIS6</td>
<td>TRIS5</td>
<td>TRIS4</td>
<td>TRIS3</td>
<td>TRIS2</td>
<td>TRIS1</td>
<td>TRIS0</td>
<td>3F6h</td>
<td>264</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FB5h</td>
<td>TRISD(2)</td>
<td>TRISD7</td>
<td>TRISD6</td>
<td>TRISD5</td>
<td>TRISD4</td>
<td>TRISD3</td>
<td>TRISD2</td>
<td>TRISD1</td>
<td>TRISD0</td>
<td>3FB6h</td>
<td>264</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FCh</td>
<td>TRISC</td>
<td>TRISC7</td>
<td>TRISC6</td>
<td>TRISC5</td>
<td>TRISC4</td>
<td>TRISC3</td>
<td>TRISC2</td>
<td>TRISC1</td>
<td>TRISC0</td>
<td>3FBb</td>
<td>264</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FChh</td>
<td>TRISB</td>
<td>TRISB7</td>
<td>TRISB6</td>
<td>TRISB5</td>
<td>TRISB4</td>
<td>TRISB3</td>
<td>TRISB2</td>
<td>TRISB1</td>
<td>TRISB0</td>
<td>3FBh</td>
<td>264</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FC2h</td>
<td>TRISA</td>
<td>TRISA7</td>
<td>TRISA6</td>
<td>TRISA5</td>
<td>TRISA4</td>
<td>TRISA3</td>
<td>TRISA2</td>
<td>TRISA1</td>
<td>TRISA0</td>
<td>3FBi</td>
<td>264</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FChh - 3FC0h</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FBBh</td>
<td>LATF(3)</td>
<td>LATF7</td>
<td>LATF6</td>
<td>LATF5</td>
<td>LATF4</td>
<td>LATF3</td>
<td>LATF2</td>
<td>LATF1</td>
<td>LATF0</td>
<td>3FBd</td>
<td>265</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FBEh</td>
<td>LATE(3)</td>
<td>LATE7</td>
<td>LATE6</td>
<td>LATE5</td>
<td>LATE4</td>
<td>LATE3</td>
<td>LATE2</td>
<td>LATE1</td>
<td>LATE0</td>
<td>3FBBh</td>
<td>265</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FBdh</td>
<td>LATD(3)</td>
<td>LATD7</td>
<td>LATD6</td>
<td>LATD5</td>
<td>LATD4</td>
<td>LATD3</td>
<td>LATD2</td>
<td>LATD1</td>
<td>LATD0</td>
<td>3FBEh</td>
<td>265</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FBCh</td>
<td>LATIC</td>
<td>LATIC7</td>
<td>LATIC6</td>
<td>LATIC5</td>
<td>LATIC4</td>
<td>LATIC3</td>
<td>LATIC2</td>
<td>LATIC1</td>
<td>LATIC0</td>
<td>3FBc</td>
<td>265</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FBBh</td>
<td>LATB</td>
<td>LATB7</td>
<td>LATB6</td>
<td>LATB5</td>
<td>LATB4</td>
<td>LATB3</td>
<td>LATB2</td>
<td>LATB1</td>
<td>LATB0</td>
<td>3FBAh</td>
<td>265</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FBAh</td>
<td>LATA</td>
<td>LATA7</td>
<td>LATA6</td>
<td>LATA5</td>
<td>LATA4</td>
<td>LATA3</td>
<td>LATA2</td>
<td>LATA1</td>
<td>LATA0</td>
<td>3FBBh</td>
<td>265</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FBh</td>
<td>T0CON1</td>
<td>CS&lt;2:0&gt;</td>
<td>ASYNC</td>
<td>CKPS&lt;3:0&gt;</td>
<td>3FBb</td>
<td>302</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FBh</td>
<td>T0CON0</td>
<td>EN</td>
<td>—</td>
<td>OUT</td>
<td>MD16</td>
<td>OUTPS</td>
<td>3FBa</td>
<td>301</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FB7h</td>
<td>TMR0H</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FB6h</td>
<td>TMR0L</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FB5h</td>
<td>T1CLK</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FB4h</td>
<td>T1GATE</td>
<td>GSS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FB3h</td>
<td>T1GCON</td>
<td>GE</td>
<td>GPOL</td>
<td>GTM</td>
<td>GSPM</td>
<td>GGO</td>
<td>GVAL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FB2h</td>
<td>T1CON</td>
<td>—</td>
<td>—</td>
<td>CKPS&lt;1:0&gt;</td>
<td>—</td>
<td>SYNC</td>
<td>RD16</td>
<td>ON</td>
<td>3FB1</td>
<td>338</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FB1h</td>
<td>TMR1H</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FB0h</td>
<td>TMR1L</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FAFh</td>
<td>T2RST</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FAEh</td>
<td>T2CLK</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FADh</td>
<td>T2HLT</td>
<td>PSYNC</td>
<td>CKPOL</td>
<td>CKSYNC</td>
<td>—</td>
<td>MODE</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FACH</td>
<td>T2CON</td>
<td>ON</td>
<td>—</td>
<td>CKPS</td>
<td>—</td>
<td>OUTPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FABh</td>
<td>T2PR</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FAAh</td>
<td>T2TMR</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FAAh</td>
<td>T3CLK</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FAEh</td>
<td>T3GATE</td>
<td>GSS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FA7h</td>
<td>T3GCON</td>
<td>GE</td>
<td>GPOL</td>
<td>GTM</td>
<td>GSPM</td>
<td>GGO</td>
<td>GVAL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FA6h</td>
<td>T3CON</td>
<td>—</td>
<td>—</td>
<td>CKPS</td>
<td>—</td>
<td>NOT_SYNC</td>
<td>RD16</td>
<td>ON</td>
<td>3FA5h</td>
<td>338</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FA5h</td>
<td>TMR3H</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FA4h</td>
<td>TMR3L</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FA3h</td>
<td>T4RST</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FA2h</td>
<td>T4CLK</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### TABLE 42-1: REGISTER FILE SUMMARY FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3FA1h</td>
<td>T4HLT</td>
<td>PSYNC</td>
<td>CKPOL</td>
<td>CKSYNC</td>
<td>MODE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>339</td>
</tr>
<tr>
<td>3FA0h</td>
<td>T4CON</td>
<td></td>
<td>CKPS</td>
<td></td>
<td>OUTPS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>338</td>
</tr>
<tr>
<td>3F9Fh</td>
<td>T4PR</td>
<td></td>
<td></td>
<td></td>
<td>PR4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>337</td>
</tr>
<tr>
<td>3F9Eh</td>
<td>T4TMR</td>
<td></td>
<td></td>
<td></td>
<td>TMR4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>337</td>
</tr>
<tr>
<td>3F9Dh</td>
<td>T5CLK</td>
<td></td>
<td></td>
<td></td>
<td>CS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>335</td>
</tr>
<tr>
<td>3F9Ch</td>
<td>T5GATE</td>
<td></td>
<td></td>
<td></td>
<td>GSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>316</td>
</tr>
<tr>
<td>3F9Bh</td>
<td>T5GCON</td>
<td>GE</td>
<td>GPOL</td>
<td>GTM</td>
<td>GSPM</td>
<td>GGO</td>
<td>GVAL</td>
<td></td>
<td></td>
<td>314</td>
</tr>
<tr>
<td>3F9Ah</td>
<td>T5CON</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CKPS</td>
<td>—</td>
<td>NOT_SYNC</td>
<td>RD16</td>
<td>ON</td>
<td>338</td>
</tr>
<tr>
<td>3F99h</td>
<td>TMR5H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TMR5H</td>
<td></td>
<td></td>
<td>317</td>
</tr>
<tr>
<td>3F98h</td>
<td>TMR5L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TMR5L</td>
<td></td>
<td></td>
<td>317</td>
</tr>
<tr>
<td>3F97h</td>
<td>T6RST</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>RSEL</td>
<td></td>
<td></td>
<td></td>
<td>336</td>
</tr>
<tr>
<td>3F96h</td>
<td>T6CLK</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CS</td>
<td></td>
<td></td>
<td></td>
<td>315</td>
</tr>
<tr>
<td>3F95h</td>
<td>T6HLT</td>
<td>PSYNC</td>
<td>CKPOL</td>
<td>CKSYNC</td>
<td>MODE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>339</td>
</tr>
<tr>
<td>3F94h</td>
<td>T6CON</td>
<td>ON</td>
<td>CKPS</td>
<td></td>
<td>OUTPS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>338</td>
</tr>
<tr>
<td>3F93h</td>
<td>T6PR</td>
<td></td>
<td></td>
<td></td>
<td>PR6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>337</td>
</tr>
<tr>
<td>3F92h</td>
<td>T6TMR</td>
<td></td>
<td></td>
<td></td>
<td>TMR6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>337</td>
</tr>
<tr>
<td>3F91h -</td>
<td></td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
<td></td>
<td></td>
<td>338</td>
</tr>
<tr>
<td>3F80h</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unimplemented</td>
<td></td>
<td></td>
<td>338</td>
</tr>
<tr>
<td>3F7Fh</td>
<td>CCP1CAP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CTS</td>
<td></td>
<td></td>
<td></td>
<td>352</td>
</tr>
<tr>
<td>3F7Eh</td>
<td>CCP1CON</td>
<td>EN</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>FMT</td>
<td>—</td>
<td>MODE</td>
<td></td>
<td>350</td>
</tr>
<tr>
<td>3F7Oh</td>
<td>CCPR1H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RH</td>
<td></td>
<td></td>
<td></td>
<td>353</td>
</tr>
<tr>
<td>3F7Ch</td>
<td>CCPR1L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RL</td>
<td></td>
<td></td>
<td></td>
<td>352</td>
</tr>
<tr>
<td>3F78h</td>
<td>CCP2CAP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CTS</td>
<td></td>
<td></td>
<td></td>
<td>352</td>
</tr>
<tr>
<td>3F7Ah</td>
<td>CCP2CON</td>
<td>EN</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>FMT</td>
<td>—</td>
<td>MODE</td>
<td></td>
<td>350</td>
</tr>
<tr>
<td>3F79h</td>
<td>CCPR2H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RH</td>
<td></td>
<td></td>
<td></td>
<td>353</td>
</tr>
<tr>
<td>3F78h</td>
<td>CCPR2L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RL</td>
<td></td>
<td></td>
<td></td>
<td>352</td>
</tr>
<tr>
<td>3F77h</td>
<td>CCP3CAP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CTS</td>
<td></td>
<td></td>
<td></td>
<td>352</td>
</tr>
<tr>
<td>3F76h</td>
<td>CCP3CON</td>
<td>EN</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>FMT</td>
<td>—</td>
<td>MODE</td>
<td></td>
<td>350</td>
</tr>
<tr>
<td>3F75h</td>
<td>CCPR3H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RH</td>
<td></td>
<td></td>
<td></td>
<td>353</td>
</tr>
<tr>
<td>3F74h</td>
<td>CCPR3L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RL</td>
<td></td>
<td></td>
<td></td>
<td>352</td>
</tr>
<tr>
<td>3F73h</td>
<td>CCP4CAP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CTS</td>
<td></td>
<td></td>
<td></td>
<td>352</td>
</tr>
<tr>
<td>3F72h</td>
<td>CCP4CON</td>
<td>EN</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>FMT</td>
<td>—</td>
<td>MODE</td>
<td></td>
<td>350</td>
</tr>
<tr>
<td>3F71h</td>
<td>CCPR4H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RH</td>
<td></td>
<td></td>
<td></td>
<td>353</td>
</tr>
<tr>
<td>3F70h</td>
<td>CCPR4L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RL</td>
<td></td>
<td></td>
<td></td>
<td>352</td>
</tr>
<tr>
<td>3F6Fh</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>3F6Eh</td>
<td>PWM5CON</td>
<td>EN</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>358</td>
</tr>
<tr>
<td>3F6Dh</td>
<td>PWM5DCH</td>
<td>DC9</td>
<td>DC8</td>
<td>DC7</td>
<td>DC6</td>
<td>DC5</td>
<td>DC4</td>
<td>DC3</td>
<td>DC2</td>
<td>360</td>
</tr>
<tr>
<td>3F6Dh</td>
<td>PWM5DCH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>360</td>
</tr>
<tr>
<td>3F6Ch</td>
<td>PWM5DCL</td>
<td>DC1</td>
<td>DC0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>360</td>
</tr>
<tr>
<td>3F6Ch</td>
<td>PWM5DCL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>360</td>
</tr>
<tr>
<td>3F6Bh</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>3F6Ah</td>
<td>PWM6CON</td>
<td>EN</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>358</td>
</tr>
<tr>
<td>3F69h</td>
<td>PWM6DCH</td>
<td>DC9</td>
<td>DC7</td>
<td>DC6</td>
<td>DC5</td>
<td>DC4</td>
<td>DC3</td>
<td>DC2</td>
<td>DC1</td>
<td>360</td>
</tr>
<tr>
<td>3F69h</td>
<td>PWM6DCH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>360</td>
</tr>
<tr>
<td>3F68h</td>
<td>PWM6DCL</td>
<td>DC1</td>
<td>DC0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>360</td>
</tr>
<tr>
<td>3F68h</td>
<td>PWM6DCL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>360</td>
</tr>
<tr>
<td>3F67h</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
</tr>
</tbody>
</table>

**Legend:**
- x = unknown
- u = unchanged
- — = unimplemented
- q = value depends on condition

**Note:**
1: Unimplemented in LF devices.
2: Unimplemented in PIC18(L)F26/27K42.
4: Unimplemented in PIC18(L)F45/55K42.
## TABLE 42-1: REGISTER FILE SUMMARY FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3F66h</td>
<td>PWM7CON</td>
<td>EN</td>
<td>—</td>
<td>OUT</td>
<td>POL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>358</td>
</tr>
<tr>
<td>3F65h</td>
<td>PWM7DCH</td>
<td>DC9</td>
<td>DC8</td>
<td>DC7</td>
<td>DC6</td>
<td>DC5</td>
<td>DC4</td>
<td>DC3</td>
<td>DC2</td>
<td>360</td>
</tr>
<tr>
<td>3F65h</td>
<td>PWM7DCH</td>
<td>DC</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>360</td>
</tr>
<tr>
<td>3F64h</td>
<td>PWM7DCL</td>
<td>DC1</td>
<td>DC0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>360</td>
</tr>
<tr>
<td>3F63h</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>360</td>
</tr>
<tr>
<td>3F62h</td>
<td>PWM8CON</td>
<td>EN</td>
<td>—</td>
<td>OUT</td>
<td>POL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>358</td>
</tr>
<tr>
<td>3F61h</td>
<td>PWM8DCH</td>
<td>DC9</td>
<td>DC8</td>
<td>DC7</td>
<td>DC6</td>
<td>DC5</td>
<td>DC4</td>
<td>DC3</td>
<td>DC2</td>
<td>360</td>
</tr>
<tr>
<td>3F61h</td>
<td>PWM8DCH</td>
<td>DC</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>360</td>
</tr>
<tr>
<td>3F60h</td>
<td>PWM8DCL</td>
<td>DC1</td>
<td>DC0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>360</td>
</tr>
<tr>
<td>3F60h</td>
<td>PWM8DCL</td>
<td>DC</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>360</td>
</tr>
<tr>
<td>3F5Fh</td>
<td>CCPTMRS1</td>
<td>P8TSEL</td>
<td>P7TSEL</td>
<td>P6TSEL</td>
<td>P5TSEL</td>
<td>359</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3F5Eh</td>
<td>CCPTMRS0</td>
<td>C4TSEL</td>
<td>C3TSEL</td>
<td>C2TSEL</td>
<td>C1TSEL</td>
<td>359</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3F5Dh</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>359</td>
</tr>
<tr>
<td>3F5Ch</td>
<td>CWG1STR</td>
<td>OVRD</td>
<td>OVRC</td>
<td>OVRB</td>
<td>OVRA</td>
<td>STRD</td>
<td>STRC</td>
<td>STRB</td>
<td>STRA</td>
<td>428</td>
</tr>
<tr>
<td>3F5Bh</td>
<td>CWG1AS1</td>
<td>—</td>
<td>AS6E</td>
<td>AS6E</td>
<td>AS4E</td>
<td>AS3E</td>
<td>AS2E</td>
<td>AS1E</td>
<td>AS0E</td>
<td>430</td>
</tr>
<tr>
<td>3F5Ah</td>
<td>CWG1AS0</td>
<td>SHUTDOWN</td>
<td>REN</td>
<td>LSBD</td>
<td>LSAC</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>429</td>
</tr>
<tr>
<td>3F57h</td>
<td>CWG1CON1</td>
<td>—</td>
<td>—</td>
<td>IN</td>
<td>—</td>
<td>POLD</td>
<td>POLC</td>
<td>POLB</td>
<td>POLA</td>
<td>425</td>
</tr>
<tr>
<td>3F56h</td>
<td>CWG1CON0</td>
<td>EN</td>
<td>LD</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>424</td>
</tr>
<tr>
<td>3F55h</td>
<td>CWG1DBF</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>431</td>
</tr>
<tr>
<td>3F54h</td>
<td>CWG1DBR</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>431</td>
</tr>
<tr>
<td>3F53h</td>
<td>CWG1ISM</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>427</td>
</tr>
<tr>
<td>3F52h</td>
<td>CWG1CLK</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>426</td>
</tr>
<tr>
<td>3F51h</td>
<td>CWG2STR</td>
<td>OVRD</td>
<td>OVRC</td>
<td>OVRB</td>
<td>OVRA</td>
<td>STRD</td>
<td>STRC</td>
<td>STRB</td>
<td>STRA</td>
<td>428</td>
</tr>
<tr>
<td>3F50h</td>
<td>CWG2AS1</td>
<td>—</td>
<td>AS6E</td>
<td>AS6E</td>
<td>AS4E</td>
<td>AS3E</td>
<td>AS2E</td>
<td>AS1E</td>
<td>AS0E</td>
<td>430</td>
</tr>
<tr>
<td>3F4Fh</td>
<td>CWG2AS0</td>
<td>SHUTDOWN</td>
<td>REN</td>
<td>LSBD</td>
<td>LSAC</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>429</td>
</tr>
<tr>
<td>3F4Eh</td>
<td>CWG2CON1</td>
<td>—</td>
<td>—</td>
<td>IN</td>
<td>—</td>
<td>POLD</td>
<td>POLC</td>
<td>POLB</td>
<td>POLA</td>
<td>425</td>
</tr>
<tr>
<td>3F4Dh</td>
<td>CWG2CON0</td>
<td>EN</td>
<td>LD</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>424</td>
</tr>
<tr>
<td>3F4Ch</td>
<td>CWG2DBF</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>431</td>
</tr>
<tr>
<td>3F4Bh</td>
<td>CWG2DBR</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>431</td>
</tr>
<tr>
<td>3F4Ah</td>
<td>CWG2ISM</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>427</td>
</tr>
<tr>
<td>3F49h</td>
<td>CWG2CLK</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>426</td>
</tr>
<tr>
<td>3F48h</td>
<td>CWG3STR</td>
<td>OVRD</td>
<td>OVRC</td>
<td>OVRB</td>
<td>OVRA</td>
<td>STRD</td>
<td>STRC</td>
<td>STRB</td>
<td>STRA</td>
<td>428</td>
</tr>
<tr>
<td>3F47h</td>
<td>CWG3AS1</td>
<td>—</td>
<td>AS6E</td>
<td>AS6E</td>
<td>AS4E</td>
<td>AS3E</td>
<td>AS2E</td>
<td>AS1E</td>
<td>AS0E</td>
<td>430</td>
</tr>
<tr>
<td>3F46h</td>
<td>CWG3AS0</td>
<td>SHUTDOWN</td>
<td>REN</td>
<td>LSBD</td>
<td>LSAC</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>429</td>
</tr>
<tr>
<td>3F45h</td>
<td>CWG3CON1</td>
<td>—</td>
<td>—</td>
<td>IN</td>
<td>—</td>
<td>POLD</td>
<td>POLC</td>
<td>POLB</td>
<td>POLA</td>
<td>425</td>
</tr>
<tr>
<td>3F44h</td>
<td>CWG3CON0</td>
<td>EN</td>
<td>LD</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>424</td>
</tr>
<tr>
<td>3F43h</td>
<td>CWG3DBF</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>431</td>
</tr>
<tr>
<td>3F42h</td>
<td>CWG3DBR</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>431</td>
</tr>
<tr>
<td>3F41h</td>
<td>CWG3ISM</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>427</td>
</tr>
<tr>
<td>3F3Fh</td>
<td>NCO1CLK</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>426</td>
</tr>
<tr>
<td>3F3Eh</td>
<td>NCO1CON</td>
<td>EN</td>
<td>—</td>
<td>OUT</td>
<td>POL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>453</td>
</tr>
<tr>
<td>3F3Dh</td>
<td>NCO1INC</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>457</td>
</tr>
<tr>
<td>3F3Ch</td>
<td>NCO1INCH</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>456</td>
</tr>
<tr>
<td>3F3Bh</td>
<td>NCO1INCL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>456</td>
</tr>
</tbody>
</table>

**Legend:**  
- x = unknown, u = unchanged, — = unimplemented, q = value depends on condition  

**Note:**  
1: Unimplemented in LF devices.  
2: Unimplemented in PIC18(L)F28/27K42.  
4: Unimplemented in PIC18(L)F45/55K42.  

© 2017 Microchip Technology Inc.  
Prepared in-house by Microchip Technology Inc.
### TABLE 42-1: REGISTER FILE SUMMARY FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3F3Ah</td>
<td>NCO1ACCU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>456</td>
</tr>
<tr>
<td>3F39h</td>
<td>NCO1ACCH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>455</td>
</tr>
<tr>
<td>3F38h</td>
<td>NCO1ACCL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>455</td>
</tr>
<tr>
<td>3F37h</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unimplemented</td>
</tr>
<tr>
<td>3F23h</td>
<td>SMT1WIN</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>398</td>
</tr>
<tr>
<td>3F22h</td>
<td>SMT1SIG</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>399</td>
</tr>
<tr>
<td>3F21h</td>
<td>SMT1CLK</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>397</td>
</tr>
<tr>
<td>3F20h</td>
<td>SMT1STAT</td>
<td>CPRUP</td>
<td>CPWUP</td>
<td>RST</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>TS</td>
<td>—</td>
<td>396</td>
</tr>
<tr>
<td>3F1Fh</td>
<td>SMT1CON1</td>
<td>GO</td>
<td>REPEAT</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>395</td>
</tr>
<tr>
<td>3F1Eh</td>
<td>SMT1CON0</td>
<td>EN</td>
<td>—</td>
<td>STP</td>
<td>WPOL</td>
<td>SPOL</td>
<td>CPOL</td>
<td>PS</td>
<td>—</td>
<td>394</td>
</tr>
<tr>
<td>3F1Dh</td>
<td>SMT1PRU</td>
<td>PR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>403</td>
</tr>
<tr>
<td>3F1Ch</td>
<td>SMT1PRH</td>
<td>PR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>403</td>
</tr>
<tr>
<td>3F19h</td>
<td>SMT1PRL</td>
<td>PR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>403</td>
</tr>
<tr>
<td>3F1Ah</td>
<td>SMT1CPWU</td>
<td>CPW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>402</td>
</tr>
<tr>
<td>3F19h</td>
<td>SMT1CPWH</td>
<td>CPW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>402</td>
</tr>
<tr>
<td>3F18h</td>
<td>SMT1CPWL</td>
<td>CPW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>402</td>
</tr>
<tr>
<td>3F17h</td>
<td>SMT1CPRU</td>
<td>CPR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>401</td>
</tr>
<tr>
<td>3F16h</td>
<td>SMT1CPRH</td>
<td>CPR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>401</td>
</tr>
<tr>
<td>3F15h</td>
<td>SMT1CPRL</td>
<td>CPR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>401</td>
</tr>
<tr>
<td>3F14h</td>
<td>SMT1TMRU</td>
<td>TMR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>400</td>
</tr>
<tr>
<td>3F13h</td>
<td>SMT1TMRH</td>
<td>TMR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>400</td>
</tr>
<tr>
<td>3F12h</td>
<td>SMT1TMRL</td>
<td>TMR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>400</td>
</tr>
<tr>
<td>3F11h</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unimplemented</td>
</tr>
<tr>
<td>3EFFh</td>
<td>ADCLK</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>622</td>
</tr>
<tr>
<td>3EFeh</td>
<td>ADACT</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>635</td>
</tr>
<tr>
<td>3EFdh</td>
<td>ADREF</td>
<td>NREF</td>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>622</td>
</tr>
<tr>
<td>3EFCh</td>
<td>ADSTAT</td>
<td>OV</td>
<td>UTHR</td>
<td>LTHR</td>
<td>MATH</td>
<td>—</td>
<td>—</td>
<td>STAT</td>
<td>—</td>
<td>621</td>
</tr>
<tr>
<td>3EFbh</td>
<td>ADCON3</td>
<td>PSIS</td>
<td>SOI</td>
<td>TMD</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>620</td>
</tr>
<tr>
<td>3EFah</td>
<td>ADCON2</td>
<td>PSIS</td>
<td>CRS</td>
<td>ACLR</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>619</td>
</tr>
<tr>
<td>3EF9h</td>
<td>ADCON1</td>
<td>PPOL</td>
<td>IPEN</td>
<td>GPOL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>618</td>
</tr>
<tr>
<td>3EF8h</td>
<td>ADCON0</td>
<td>ON</td>
<td>CONT</td>
<td>—</td>
<td>CS</td>
<td>FM</td>
<td>—</td>
<td>GO</td>
<td>—</td>
<td>617</td>
</tr>
<tr>
<td>3EF7h</td>
<td>ADPREH</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>624</td>
</tr>
<tr>
<td>3EF6h</td>
<td>ADPREL</td>
<td>PRE</td>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>624</td>
</tr>
<tr>
<td>3EF5h</td>
<td>ADCAP</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ADCAP</td>
<td>—</td>
<td>626</td>
</tr>
<tr>
<td>3EF4h</td>
<td>ADACOH</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ACQ</td>
<td>—</td>
<td>625</td>
</tr>
<tr>
<td>3EF3h</td>
<td>ADACOL</td>
<td>ACQ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>625</td>
</tr>
<tr>
<td>3EF2h</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unimplemented</td>
</tr>
<tr>
<td>3EF1h</td>
<td>ADPCH</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>623</td>
</tr>
<tr>
<td>3EF0h</td>
<td>ADRESH</td>
<td>RES</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>628</td>
</tr>
<tr>
<td>3EEFh</td>
<td>ADRESL</td>
<td>RES</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>628</td>
</tr>
<tr>
<td>3EEDh</td>
<td>ADPRESH</td>
<td>RES</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>628</td>
</tr>
<tr>
<td>3EECh</td>
<td>ADPREVH</td>
<td>RES</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>628</td>
</tr>
<tr>
<td>3EECh</td>
<td>ADPREVL</td>
<td>RES</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>628</td>
</tr>
<tr>
<td>3EECh</td>
<td>ADRPT</td>
<td>RTP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>626</td>
</tr>
<tr>
<td>3EECh</td>
<td>ADCNT</td>
<td>CNT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>627</td>
</tr>
<tr>
<td>3EEAh</td>
<td>ADACCU</td>
<td>ACC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>631</td>
</tr>
<tr>
<td>3EE9h</td>
<td>ADACCH</td>
<td>ACC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>631</td>
</tr>
</tbody>
</table>

**Legend:**  
- x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

**Note:**  
1. Unimplemented in LF devices.
2. Unimplemented in PIC18(L)F26/27K42.
4. Unimplemented in PIC18(L)F45/55K42.

© 2017 Microchip Technology Inc. Preliminary
**TABLE 42-1: REGISTER FILE SUMMARY FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES**

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3EE8h</td>
<td>ADACCL</td>
<td>ACC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>631</td>
</tr>
<tr>
<td>3EE7h</td>
<td>ADFLTRH</td>
<td>FLTR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>627</td>
</tr>
<tr>
<td>3EE6h</td>
<td>ADFLTRL</td>
<td>FLTR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>627</td>
</tr>
<tr>
<td>3EE5h</td>
<td>ADSTPTH</td>
<td>STPT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>632</td>
</tr>
<tr>
<td>3EE4h</td>
<td>ADSTPTL</td>
<td>STPT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>632</td>
</tr>
<tr>
<td>3EE3h</td>
<td>ADERRH</td>
<td>ERR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>633</td>
</tr>
<tr>
<td>3EE2h</td>
<td>ADERRL</td>
<td>ERR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>633</td>
</tr>
<tr>
<td>3EE1h</td>
<td>ADUTHH</td>
<td>UTH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>634</td>
</tr>
<tr>
<td>3EE0h</td>
<td>ADUTHL</td>
<td>UTH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>634</td>
</tr>
<tr>
<td>3EDFh</td>
<td>ADLTHH</td>
<td>LTH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>633</td>
</tr>
<tr>
<td>3EDeh</td>
<td>ADLTHL</td>
<td>LTH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>634</td>
</tr>
<tr>
<td>3EDDh - 3ED6h</td>
<td>— — — — — CPRDY</td>
<td>636</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3ED7h</td>
<td>ADCP</td>
<td>ON</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>630</td>
</tr>
<tr>
<td>3ED6h - 3EC8h</td>
<td>— — — — — CPRDY</td>
<td>636</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3EC9h</td>
<td>HLVDCON1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>SEL</td>
<td>658</td>
</tr>
<tr>
<td>3EC8h</td>
<td>HLVDCON0</td>
<td>EN</td>
<td>—</td>
<td>OUT</td>
<td>RDY</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>INTL</td>
<td>657</td>
</tr>
<tr>
<td>3EC7h - 3EC4h</td>
<td>— — — — — —</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>3EC3h</td>
<td>ZCDCON</td>
<td>SEN</td>
<td>—</td>
<td>OUT</td>
<td>POL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>INTP</td>
<td>462</td>
</tr>
<tr>
<td>3EC2h</td>
<td>— — — — — — —</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>INTP</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>3EC1h</td>
<td>FVRCON</td>
<td>EN</td>
<td>RDY</td>
<td>TSEN</td>
<td>TSRNG</td>
<td>CDAFVR</td>
<td>ADFVR</td>
<td>—</td>
<td>—</td>
<td>597</td>
</tr>
<tr>
<td>3EC0h</td>
<td>CMOUT</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>C2OUT</td>
<td>650</td>
</tr>
<tr>
<td>3EBFh</td>
<td>CM1PCH</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>PCH</td>
<td>650</td>
</tr>
<tr>
<td>3EBEh</td>
<td>CM1NCH</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>NCH</td>
<td>649</td>
</tr>
<tr>
<td>3EBDh</td>
<td>CM1CON1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>INTP</td>
<td>649</td>
</tr>
<tr>
<td>3EBCh</td>
<td>CM1CON0</td>
<td>EN</td>
<td>OUT</td>
<td>—</td>
<td>POL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>HYS</td>
<td>648</td>
</tr>
<tr>
<td>3EBBh</td>
<td>CM2PCH</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>PCH</td>
<td>650</td>
</tr>
<tr>
<td>3EBAh</td>
<td>CM2NCH</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>NCH</td>
<td>649</td>
</tr>
<tr>
<td>3EB9h</td>
<td>CM2CON1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>INTP</td>
<td>649</td>
</tr>
<tr>
<td>3EB8h</td>
<td>CM2CON0</td>
<td>EN</td>
<td>OUT</td>
<td>—</td>
<td>POL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>HYS</td>
<td>648</td>
</tr>
<tr>
<td>3EB7h - 3E9Fh</td>
<td>— — — — — —</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>3E9Eh</td>
<td>DAC1CON0</td>
<td>EN</td>
<td>—</td>
<td>OE1</td>
<td>OE2</td>
<td>PSS</td>
<td>—</td>
<td>—</td>
<td>NSS</td>
<td>640</td>
</tr>
<tr>
<td>3E9Dh</td>
<td>— — — — — — —</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>3E9Ch</td>
<td>DAC1CON1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>DATA</td>
<td>641</td>
</tr>
<tr>
<td>3E9Bh - 3DF8h</td>
<td>— — — — — —</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>3DFAh</td>
<td>U1ERRIE</td>
<td>TXMTIE</td>
<td>PERIE</td>
<td>ABDOVE</td>
<td>ABDOVE</td>
<td>CERIE</td>
<td>FERIE</td>
<td>RXBIE</td>
<td>RXFIE</td>
<td>TXCIE</td>
</tr>
<tr>
<td>3DF9h</td>
<td>U1ERRIR</td>
<td>TXMTIF</td>
<td>PERIF</td>
<td>ABDORF</td>
<td>ABDOVF</td>
<td>CERIF</td>
<td>FERIF</td>
<td>RXBKF</td>
<td>RXFOIF</td>
<td>TXCIF</td>
</tr>
<tr>
<td>3DF8h</td>
<td>U1UIR</td>
<td>WUIF</td>
<td>ABDIF</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ABDIE</td>
<td>—</td>
</tr>
<tr>
<td>3DF7h</td>
<td>U1IFO</td>
<td>TXWRE</td>
<td>STPMD</td>
<td>TXBE</td>
<td>TXBF</td>
<td>TXBF</td>
<td>RXIDL</td>
<td>XON</td>
<td>RXBE</td>
<td>RXBF</td>
</tr>
<tr>
<td>3DF6h</td>
<td>U1BRGH</td>
<td>BRGH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>505</td>
</tr>
<tr>
<td>3DF5h</td>
<td>U1BRGL</td>
<td>BRGL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>505</td>
</tr>
<tr>
<td>3DF4h</td>
<td>U1CON2</td>
<td>RUNOVF</td>
<td>RXPOL</td>
<td>STP</td>
<td>C0EN</td>
<td>TXPOL</td>
<td>FLO</td>
<td></td>
<td></td>
<td>500</td>
</tr>
<tr>
<td>3DF3h</td>
<td>U1CON1</td>
<td>ON</td>
<td>—</td>
<td>—</td>
<td>WUE</td>
<td>RXBMD</td>
<td>—</td>
<td>BRKOVF</td>
<td>SENDB</td>
<td>499</td>
</tr>
<tr>
<td>3DF2h</td>
<td>U1CON0</td>
<td>BRGS</td>
<td>ABDEN</td>
<td>TXEN</td>
<td>RXEN</td>
<td>MODE</td>
<td></td>
<td></td>
<td></td>
<td>498</td>
</tr>
<tr>
<td>3DF1h</td>
<td>U1P3H</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>P3H</td>
<td>509</td>
</tr>
</tbody>
</table>

**Legend:**
- x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

**Note:**
1. Unimplemented in LF devices.
2. Unimplemented in PIC18(L)F26/27K42.
4. Unimplemented in PIC18(L)F45/55K42.
TABLE 42-1: REGISTER FILE SUMMARY FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3DF0h</td>
<td>U1P3L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P3L</td>
<td>509</td>
</tr>
<tr>
<td>3DEFh</td>
<td>U1P2H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P2H</td>
<td></td>
<td>508</td>
</tr>
<tr>
<td>3DE Eh</td>
<td>U1P2L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P2L</td>
<td></td>
<td>508</td>
</tr>
<tr>
<td>3DE Dh</td>
<td>U1P1H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P1H</td>
<td></td>
<td>507</td>
</tr>
<tr>
<td>3DE ch</td>
<td>U1P1L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P1L</td>
<td></td>
<td>507</td>
</tr>
<tr>
<td>3DE Bh</td>
<td>U1TXCHK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TXCHK</td>
<td></td>
<td>510</td>
</tr>
<tr>
<td>3DE Ah</td>
<td>U1TBX</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TXB</td>
<td></td>
<td>506</td>
</tr>
<tr>
<td>3DE9h</td>
<td>U1RXCHK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RXCHK</td>
<td></td>
<td>510</td>
</tr>
<tr>
<td>3DE8h</td>
<td>U1RXB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RXB</td>
<td></td>
<td>506</td>
</tr>
<tr>
<td>3DE7h</td>
<td></td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unimplemented</td>
</tr>
<tr>
<td>3DE6h</td>
<td></td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unimplemented</td>
</tr>
<tr>
<td>3DE5h</td>
<td></td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unimplemented</td>
</tr>
<tr>
<td>3DE4h</td>
<td></td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unimplemented</td>
</tr>
<tr>
<td>3DE3h</td>
<td></td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unimplemented</td>
</tr>
<tr>
<td>3DE2h</td>
<td>U2ERRIE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>502</td>
</tr>
<tr>
<td>3DE1h</td>
<td>U2ERRIR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>501</td>
</tr>
<tr>
<td>3DE0h</td>
<td>U2UIR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>503</td>
</tr>
<tr>
<td>3DDFh</td>
<td>U2FIFO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>504</td>
</tr>
<tr>
<td>3DDDh</td>
<td>U2BRGH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BRGH</td>
<td>505</td>
</tr>
<tr>
<td>3DDCh</td>
<td>U2BRGL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BRGL</td>
<td>505</td>
</tr>
<tr>
<td>3DD Bh</td>
<td>U2CON2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RUNOVF</td>
<td>500</td>
</tr>
<tr>
<td>3DD Ah</td>
<td>U2CON1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ON</td>
<td>499</td>
</tr>
<tr>
<td>3DD9h</td>
<td>U2CON0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BRGS</td>
<td>498</td>
</tr>
<tr>
<td>3DD8h</td>
<td>U2P3L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P3L</td>
<td></td>
<td>508</td>
</tr>
<tr>
<td>3DD7h</td>
<td></td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unimplemented</td>
</tr>
<tr>
<td>3DD6h</td>
<td>U2P2L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P2L</td>
<td></td>
<td>508</td>
</tr>
<tr>
<td>3DD5h</td>
<td></td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unimplemented</td>
</tr>
<tr>
<td>3DD4h</td>
<td>U2P1L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P1L</td>
<td></td>
<td>507</td>
</tr>
<tr>
<td>3DD3h</td>
<td></td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unimplemented</td>
</tr>
<tr>
<td>3DD2h</td>
<td>U2TXB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TXB</td>
<td></td>
<td>506</td>
</tr>
<tr>
<td>3DD1h</td>
<td></td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unimplemented</td>
</tr>
<tr>
<td>3DD0h</td>
<td>U2RXB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RXB</td>
<td></td>
<td>506</td>
</tr>
<tr>
<td>3DCh</td>
<td></td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unimplemented</td>
</tr>
<tr>
<td>3D7h</td>
<td>I2C1BTO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BTO</td>
<td></td>
<td>582</td>
</tr>
<tr>
<td>3D7Bh</td>
<td>I2C1CLK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CLK</td>
<td></td>
<td>581</td>
</tr>
<tr>
<td>3D7Ah</td>
<td>I2C1 PIE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CNTIE</td>
<td>WRIE</td>
<td>588</td>
</tr>
<tr>
<td>3D79h</td>
<td>I2C1 PIR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ACKTIE</td>
<td>ADBIE</td>
<td>587</td>
</tr>
<tr>
<td>3D78h</td>
<td>I2C1 STAT1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>RXRE</td>
<td>584</td>
</tr>
<tr>
<td>3D77h</td>
<td>I2C1 STAT0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SMA</td>
<td>D</td>
<td>583</td>
</tr>
<tr>
<td>3D76h</td>
<td>I2C1 ERR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BTOIF</td>
<td>BCLIF</td>
<td>585</td>
</tr>
<tr>
<td>3D75h</td>
<td>I2C1 CON2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ACNT</td>
<td>ABD</td>
<td>580</td>
</tr>
<tr>
<td>3D74h</td>
<td>I2C1 CON1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ACKDT</td>
<td>ACKSTAT</td>
<td>579</td>
</tr>
<tr>
<td>3D73h</td>
<td>I2C1 CON0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S</td>
<td>CSTR</td>
<td>577</td>
</tr>
<tr>
<td>3D72h</td>
<td>I2C1 AD R3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ADR</td>
<td></td>
<td>592</td>
</tr>
<tr>
<td>3D71h</td>
<td>I2C1 AD R2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td></td>
<td>591</td>
</tr>
<tr>
<td>3D70h</td>
<td>I2C1 AD R1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ADR</td>
<td></td>
<td>590</td>
</tr>
<tr>
<td>3D6Fh</td>
<td>I2C1 AD R0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ADR</td>
<td></td>
<td>589</td>
</tr>
<tr>
<td>3D6 Eh</td>
<td>I2C1 AD B1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ADB</td>
<td></td>
<td>594</td>
</tr>
<tr>
<td>3D6 Dh</td>
<td>I2C1 AD B0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ADB</td>
<td></td>
<td>593</td>
</tr>
</tbody>
</table>

Legend:  
- x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

Note:  
1: Unimplemented in LF devices.
2: Unimplemented in PIC18(L)F26/27K42.
4: Unimplemented in PIC18(L)F45/55K42.
## TABLE 42-1: REGISTER FILE SUMMARY FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D6Ch</td>
<td>I2C1CNT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>586</td>
</tr>
<tr>
<td>3D68h</td>
<td>I2C1TXB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TXB</td>
</tr>
<tr>
<td>3D6Ah</td>
<td>I2C1RXB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RXB</td>
</tr>
<tr>
<td>3D69h -</td>
<td>I2C1STAT</td>
<td></td>
<td></td>
<td>Unimplemented</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3D6Ah</td>
<td>I2C2STAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3D6Ah</td>
<td>I2C2TXB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TXB</td>
</tr>
<tr>
<td>3D6Ah</td>
<td>I2C2RXB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RXB</td>
</tr>
<tr>
<td>3D6Ah</td>
<td>I2C2STAT</td>
<td>Unimplemented</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3D6Ah</td>
<td>I2C2TXB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TXB</td>
</tr>
<tr>
<td>3D6Ah</td>
<td>I2C2RXB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RXB</td>
</tr>
<tr>
<td>3D6Ah</td>
<td>I2C2STAT</td>
<td>Unimplemented</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3D6Ah</td>
<td>I2C2TXB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TXB</td>
</tr>
<tr>
<td>3D6Ah</td>
<td>I2C2RXB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RXB</td>
</tr>
<tr>
<td>3D6Ah</td>
<td>I2C2STAT</td>
<td>Unimplemented</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3D6Ah</td>
<td>I2C2TXB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TXB</td>
</tr>
<tr>
<td>3D6Ah</td>
<td>I2C2RXB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RXB</td>
</tr>
<tr>
<td>3D6Ah</td>
<td>I2C2STAT</td>
<td>Unimplemented</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3D6Ah</td>
<td>I2C2TXB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TXB</td>
</tr>
<tr>
<td>3D6Ah</td>
<td>I2C2RXB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RXB</td>
</tr>
<tr>
<td>3D6Ah</td>
<td>I2C2STAT</td>
<td>Unimplemented</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3D6Ah</td>
<td>I2C2TXB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TXB</td>
</tr>
<tr>
<td>3D6Ah</td>
<td>I2C2RXB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RXB</td>
</tr>
<tr>
<td>3D6Ah</td>
<td>I2C2STAT</td>
<td>Unimplemented</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3D6Ah</td>
<td>I2C2TXB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TXB</td>
</tr>
<tr>
<td>3D6Ah</td>
<td>I2C2RXB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RXB</td>
</tr>
<tr>
<td>3D6Ah</td>
<td>I2C2STAT</td>
<td>Unimplemented</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3D6Ah</td>
<td>I2C2TXB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TXB</td>
</tr>
<tr>
<td>3D6Ah</td>
<td>I2C2RXB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RXB</td>
</tr>
<tr>
<td>3D6Ah</td>
<td>I2C2STAT</td>
<td>Unimplemented</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3D6Ah</td>
<td>I2C2TXB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TXB</td>
</tr>
<tr>
<td>3D6Ah</td>
<td>I2C2RXB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RXB</td>
</tr>
</tbody>
</table>

### Legend:
- **x** = unknown, **u** = unchanged, **—** = unimplemented, **q** = value depends on condition

### Note:
1: Unimplemented in LF devices.
2: Unimplemented in PIC18(L)F28/27K42.
4: Unimplemented in PIC18(L)F45/55K42.
### TABLE 42-1: REGISTER FILE SUMMARY FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3C5Eh</td>
<td>CLKRCON</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3C5Fh</td>
<td>CLKRCLK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3C5Ah</td>
<td>CLC1GLS0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3C5Bh</td>
<td>CLC1GLS1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3C5Ch</td>
<td>CLC1GLS2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3C5Dh</td>
<td>CLC1GLS3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3C5Eh</td>
<td>CLC1CON</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3C5Fh</td>
<td>CLC2GLS0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3C5Bh</td>
<td>CLC2GLS1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3C5Ch</td>
<td>CLC2GLS2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3C5Dh</td>
<td>CLC2GLS3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3C5Eh</td>
<td>CLC3GLS0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3C5Fh</td>
<td>CLC3GLS1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3C5Bh</td>
<td>CLC3GLS2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3C5Ch</td>
<td>CLC3GLS3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3C5Eh</td>
<td>CLC4GLS0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3C5Fh</td>
<td>CLC4GLS1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3C5Bh</td>
<td>CLC4GLS2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3C5Ch</td>
<td>CLC4GLS3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3C60h</td>
<td>DMA1SIRQ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:  
- x = unknown, u = unchanged, — = unimplemented, q = value depends on condition  
Note:  
1: Unimplemented in LF devices.  
2: Unimplemented in PIC18(L)F26/27K42.  
4: Unimplemented in PIC18(L)F45/55K42.
### TABLE 42-1: REGISTER FILE SUMMARY FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3BFeh</td>
<td>DMA1AIRQ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>256</td>
</tr>
<tr>
<td>3BFdh</td>
<td>DMA1CON1</td>
<td>EN</td>
<td>SIRQEN</td>
<td>DGO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>249</td>
</tr>
<tr>
<td>3BFch</td>
<td>DMA1CON0</td>
<td>DMODE</td>
<td>DSTP</td>
<td>SMR</td>
<td>SMODE</td>
<td>SSTP</td>
<td></td>
<td></td>
<td></td>
<td>248</td>
</tr>
<tr>
<td>3BFb</td>
<td>DMA1SSAU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SSA</td>
<td>251</td>
</tr>
<tr>
<td>3BFah</td>
<td>DMA1SSAH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SSA</td>
<td>250</td>
</tr>
<tr>
<td>3BFg</td>
<td>DMA1SSAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SSA</td>
<td>250</td>
</tr>
<tr>
<td>3BFh</td>
<td>DMA1SSZH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SSZ</td>
<td>252</td>
</tr>
<tr>
<td>3BFg</td>
<td>DMA1SSSL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SSZ</td>
<td>252</td>
</tr>
<tr>
<td>3BFg</td>
<td>DMA1SPTRU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SPTR</td>
<td>252</td>
</tr>
<tr>
<td>3BFg</td>
<td>DMA1SPTRH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SPTR</td>
<td>251</td>
</tr>
<tr>
<td>3BFh</td>
<td>DMA1SPTRL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SPTR</td>
<td>251</td>
</tr>
<tr>
<td>3BFh</td>
<td>DMA1SCNTL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SCNT</td>
<td>253</td>
</tr>
<tr>
<td>3BFh</td>
<td>DMA1DSAH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DSA</td>
<td>254</td>
</tr>
<tr>
<td>3BFh</td>
<td>DMA1DSAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SSA</td>
<td>253</td>
</tr>
<tr>
<td>3BFh</td>
<td>DMA1DSZH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DSZ</td>
<td>255</td>
</tr>
<tr>
<td>3BFh</td>
<td>DMA1DSZL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DSZ</td>
<td>255</td>
</tr>
<tr>
<td>3BEh</td>
<td>DMA1DPTRH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DPTR</td>
<td>254</td>
</tr>
<tr>
<td>3BEh</td>
<td>DMA1DPTRL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DPTR</td>
<td>254</td>
</tr>
<tr>
<td>3BEh</td>
<td>DMA1DCNTH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DCNT</td>
<td>256</td>
</tr>
<tr>
<td>3BEh</td>
<td>DMA1DCNTL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DCNT</td>
<td>255</td>
</tr>
<tr>
<td>3BEh</td>
<td>DMA1BUFF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BUF</td>
<td>250</td>
</tr>
<tr>
<td>3BEh</td>
<td>DMA1BUFF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unimplemented</td>
<td>250</td>
</tr>
<tr>
<td>3BEh</td>
<td>DMA1BUFF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unimplemented</td>
<td>250</td>
</tr>
<tr>
<td>3Bdh</td>
<td>DMA2AIRQ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SIRQ</td>
<td>256</td>
</tr>
<tr>
<td>3Bdh</td>
<td>DMA2AIRQ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>AIRQ</td>
<td>256</td>
</tr>
<tr>
<td>3Bdh</td>
<td>DMA2CON1</td>
<td>EN</td>
<td>SIRQEN</td>
<td>DGO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>249</td>
</tr>
<tr>
<td>3Bdh</td>
<td>DMA2CON0</td>
<td>DMODE</td>
<td>DSTP</td>
<td>SMR</td>
<td>SMODE</td>
<td>SSTP</td>
<td></td>
<td></td>
<td></td>
<td>248</td>
</tr>
<tr>
<td>3Bdh</td>
<td>DMA2SSAU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SSA</td>
<td>251</td>
</tr>
<tr>
<td>3Bdh</td>
<td>DMA2SSAH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SSA</td>
<td>250</td>
</tr>
<tr>
<td>3Bdh</td>
<td>DMA2SSAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SSA</td>
<td>250</td>
</tr>
<tr>
<td>3Bdh</td>
<td>DMA2SSZH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SSZ</td>
<td>252</td>
</tr>
<tr>
<td>3Bdh</td>
<td>DMA2SSSL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SSZ</td>
<td>252</td>
</tr>
<tr>
<td>3Bdh</td>
<td>DMA2SPTRU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SPTR</td>
<td>252</td>
</tr>
<tr>
<td>3Bdh</td>
<td>DMA2SPTRH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SPTR</td>
<td>251</td>
</tr>
<tr>
<td>3Bdh</td>
<td>DMA2SPTRL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SPTR</td>
<td>251</td>
</tr>
<tr>
<td>3Bdh</td>
<td>DMA2SCNTL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SCNT</td>
<td>253</td>
</tr>
<tr>
<td>3Bdh</td>
<td>DMA2DCNTH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DCNT</td>
<td>256</td>
</tr>
<tr>
<td>3Bdh</td>
<td>DMA2DCNTL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DCNT</td>
<td>255</td>
</tr>
<tr>
<td>3Bdh</td>
<td>DMA2BUFF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BUF</td>
<td>250</td>
</tr>
</tbody>
</table>

**Legend:**
- `x` = unknown,
- `u` = unchanged,
- `—` = unimplemented,
- `q` = value depends on condition

**Note:**
1: Unimplemented in LF devices.
2: Unimplemented in PIC18(L)F26/27K42.
4: Unimplemented in PIC18(L)F45/55K42.
### TABLE 42-1: REGISTER FILE SUMMARY FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3BC8h - 3AE8h</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>3AE8h</td>
<td>U2CTSPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>U2CTSPPS 277</td>
</tr>
<tr>
<td>3AE7h</td>
<td>U1RTXPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>U1RTXPPS 277</td>
</tr>
<tr>
<td>3AE4h</td>
<td>I2C2SDAPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>I2C2SDAPPS 277</td>
</tr>
<tr>
<td>3AE3h</td>
<td>I2C2SLPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>I2C2SLPPS 277</td>
</tr>
<tr>
<td>3AE2h</td>
<td>I2C1SDAPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>I2C1SDAPPS 277</td>
</tr>
<tr>
<td>3AE1h</td>
<td>I2C1SLPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>I2C1SLPPS 277</td>
</tr>
<tr>
<td>3AE0h</td>
<td>SPI1SDPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>SPI1SDPPS 277</td>
</tr>
<tr>
<td>3ADFh</td>
<td>SPI1SDIIPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>SPI1SDIIPPS 277</td>
</tr>
<tr>
<td>3ADEh</td>
<td>SPI1SCKPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>SPI1SCKPPS 277</td>
</tr>
<tr>
<td>3ADDh</td>
<td>ADACTPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ADACTPPS 277</td>
</tr>
<tr>
<td>3ADCh</td>
<td>CLCIN3PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CLCIN3PPS 277</td>
</tr>
<tr>
<td>3AD8h</td>
<td>CLCIN2PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CLCIN2PPS 277</td>
</tr>
<tr>
<td>3ADAh</td>
<td>CLCIN1PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CLCIN1PPS 277</td>
</tr>
<tr>
<td>3AD9h</td>
<td>CLCIN0PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CLCIN0PPS 277</td>
</tr>
<tr>
<td>3AD8h</td>
<td>MD1SRCPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>MD1SRCPPS 277</td>
</tr>
<tr>
<td>3AD7h</td>
<td>MD1CARHPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>MD1CARHPPS 277</td>
</tr>
<tr>
<td>3AD6h</td>
<td>MD1CARLPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>MD1CARLPPS 277</td>
</tr>
<tr>
<td>3AD5h</td>
<td>CW3INPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CW3INPPS 277</td>
</tr>
<tr>
<td>3AD4h</td>
<td>CW2INPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CW2INPPS 277</td>
</tr>
<tr>
<td>3AD3h</td>
<td>CW1INPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CW1INPPS 277</td>
</tr>
<tr>
<td>3AD2h</td>
<td>SMT1SIGPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>SMT1SIGPPS 277</td>
</tr>
<tr>
<td>3AD1h</td>
<td>SMT1WINPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>SMT1WINPPS 277</td>
</tr>
<tr>
<td>3ADOh</td>
<td>CCP4PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CCP4PPS 277</td>
</tr>
<tr>
<td>3ACFh</td>
<td>CCP3PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CCP3PPS 277</td>
</tr>
<tr>
<td>3ACEh</td>
<td>CCP2PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CCP2PPS 277</td>
</tr>
<tr>
<td>3ACDh</td>
<td>CCP1PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CCP1PPS 277</td>
</tr>
<tr>
<td>3ACC0h</td>
<td>T6INPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>T6INPPS 277</td>
</tr>
<tr>
<td>3ACBh</td>
<td>T4INPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>T4INPPS 277</td>
</tr>
<tr>
<td>3ACA0h</td>
<td>T2INPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>T2INPPS 277</td>
</tr>
<tr>
<td>3AC9h</td>
<td>T5GPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>T5GPPS 277</td>
</tr>
<tr>
<td>3AC8h</td>
<td>T5CLKIPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>T5CLKIPPS 277</td>
</tr>
<tr>
<td>3AC7h</td>
<td>T3GPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>T3GPPS 277</td>
</tr>
<tr>
<td>3AC6h</td>
<td>T3CLKIPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>T3CLKIPPS 277</td>
</tr>
<tr>
<td>3AC5h</td>
<td>T1GPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>T1GPPS 277</td>
</tr>
<tr>
<td>3AC4h</td>
<td>T1CLKIPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>T1CLKIPPS 277</td>
</tr>
<tr>
<td>3AC3h</td>
<td>T0CLKIPPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>T0CLKIPPS 277</td>
</tr>
<tr>
<td>3AC2h</td>
<td>INT2PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>INT2PPS 277</td>
</tr>
<tr>
<td>3AC1h</td>
<td>INT1PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>INT1PPS 277</td>
</tr>
<tr>
<td>3AC0h</td>
<td>INT0PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>INT0PPS 277</td>
</tr>
<tr>
<td>3ABFh</td>
<td>PPSLOCK</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>PPSLOCKED 283</td>
</tr>
<tr>
<td>3ABEh</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Reserved, maintain as ‘0’</td>
</tr>
<tr>
<td>3ABDh - 3AA9h</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>3AA9h</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Reserved, maintain as ‘0’</td>
</tr>
</tbody>
</table>

**Legend:**
- \( x \) = unknown
- \( u \) = unchanged
- — = unimplemented
- \( q \) = value depends on condition

**Note:**
1. Unimplemented in LF devices.
2. Unimplemented in PIC18(L)F26/27K42.
4. Unimplemented in PIC18(L)F45/55K42.
### TABLE 42-1: REGISTER FILE SUMMARY FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3A98h</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>reserved, maintain as '0'</td>
<td>270</td>
</tr>
<tr>
<td>3A97h-3A95h</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>unimplemented</td>
<td>269</td>
</tr>
<tr>
<td>3A94h</td>
<td>INLVLF&lt;sup&gt;(3)&lt;/sup&gt;</td>
<td>INLVLF7</td>
<td>INLVLF6</td>
<td>INLVLF5</td>
<td>INLVLF4</td>
<td>INLVLF3</td>
<td>INLVLF2</td>
<td>INLVLF1</td>
<td>INLVLF0</td>
<td>270</td>
</tr>
<tr>
<td>3A93h</td>
<td>SLRCONF&lt;sup&gt;(3)&lt;/sup&gt;</td>
<td>SLRCONF7</td>
<td>SLRCONF6</td>
<td>SLRCONF5</td>
<td>SLRCONF4</td>
<td>SLRCONF3</td>
<td>SLRCONF2</td>
<td>SLRCONF1</td>
<td>SLRCONF0</td>
<td>269</td>
</tr>
<tr>
<td>3A92h</td>
<td>ODCONF&lt;sup&gt;(3)&lt;/sup&gt;</td>
<td>ODCONF7</td>
<td>ODCONF6</td>
<td>ODCONF5</td>
<td>ODCONF4</td>
<td>ODCONF3</td>
<td>ODCONF2</td>
<td>ODCONF1</td>
<td>ODCONF0</td>
<td>268</td>
</tr>
<tr>
<td>3A91h</td>
<td>WPUF&lt;sup&gt;(3)&lt;/sup&gt;</td>
<td>WPUF7</td>
<td>WPUF6</td>
<td>WPUF5</td>
<td>WPUF4</td>
<td>WPUF3</td>
<td>WPUF2</td>
<td>WPUF1</td>
<td>WPUF0</td>
<td>267</td>
</tr>
<tr>
<td>3A90h</td>
<td>ANSELF&lt;sup&gt;(3)&lt;/sup&gt;</td>
<td>ANSELF7</td>
<td>ANSELF6</td>
<td>ANSELF5</td>
<td>ANSELF4</td>
<td>ANSELF3</td>
<td>ANSELF2</td>
<td>ANSELF1</td>
<td>ANSELF0</td>
<td>266</td>
</tr>
<tr>
<td>3A8Fh-3A8Ah</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>unimplemented</td>
<td>266</td>
</tr>
<tr>
<td>3A89h</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>reserved, maintain as '0'</td>
<td>267</td>
</tr>
<tr>
<td>3A88h</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>reserved, maintain as '0'</td>
<td>267</td>
</tr>
<tr>
<td>3A87h</td>
<td>IOCEF</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>287</td>
</tr>
<tr>
<td>3A86h</td>
<td>IOCEN</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>287</td>
</tr>
<tr>
<td>3A85h</td>
<td>IOCEP</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>287</td>
</tr>
<tr>
<td>3A84h</td>
<td>INLVLE</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>INLVLE3</td>
<td>INLVLE2&lt;sup&gt;(2)&lt;/sup&gt;</td>
<td>270</td>
</tr>
<tr>
<td>3A83h</td>
<td>SLRCONE&lt;sup&gt;(2)&lt;/sup&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>SRLE&lt;sup&gt;(2)&lt;/sup&gt;</td>
<td>269</td>
</tr>
<tr>
<td>3A82h</td>
<td>ODCONE&lt;sup&gt;(2)&lt;/sup&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ODCE&lt;sup&gt;(2)&lt;/sup&gt;</td>
<td>268</td>
</tr>
<tr>
<td>3A81h</td>
<td>WPUE</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>WPUE3&lt;sup&gt;(2)&lt;/sup&gt;</td>
<td>267</td>
</tr>
<tr>
<td>3A80h</td>
<td>ANSELE&lt;sup&gt;(2)&lt;/sup&gt;</td>
<td>ANSELE7</td>
<td>ANSELE6</td>
<td>ANSELE5</td>
<td>ANSELE4</td>
<td>ANSELE3</td>
<td>ANSELE2</td>
<td>ANSELE1</td>
<td>ANSELE0</td>
<td>266</td>
</tr>
<tr>
<td>3A7Fh-3A7CH</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>unimplemented</td>
<td>266</td>
</tr>
<tr>
<td>3A78h</td>
<td>RD1I2C&lt;sup&gt;(2)&lt;/sup&gt;</td>
<td>—</td>
<td>IOCEN3</td>
<td>PU</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>TH</td>
<td>263</td>
</tr>
<tr>
<td>3A7Ah</td>
<td>RD0I2C&lt;sup&gt;(2)&lt;/sup&gt;</td>
<td>—</td>
<td>IOCEN3</td>
<td>PU</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>TH</td>
<td>263</td>
</tr>
<tr>
<td>3A79h</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>reserved, maintain as '0'</td>
<td>267</td>
</tr>
<tr>
<td>3A78h</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>reserved, maintain as '0'</td>
<td>267</td>
</tr>
<tr>
<td>3A77h-3A75h</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>unimplemented</td>
<td>266</td>
</tr>
<tr>
<td>3A74h</td>
<td>INLVLD&lt;sup&gt;(2)&lt;/sup&gt;</td>
<td>INLVLD7</td>
<td>INLVLD6</td>
<td>INLVLD5</td>
<td>INLVLD4</td>
<td>INLVLD3</td>
<td>INLVLD2</td>
<td>INLVLD1</td>
<td>INLVLD0</td>
<td>270</td>
</tr>
<tr>
<td>3A73h</td>
<td>SLRCOND&lt;sup&gt;(2)&lt;/sup&gt;</td>
<td>SRLD7</td>
<td>SRLD6</td>
<td>SRLD5</td>
<td>SRLD4</td>
<td>SRLD3</td>
<td>SRLD2</td>
<td>SRLD1</td>
<td>SRLD0</td>
<td>269</td>
</tr>
<tr>
<td>3A72h</td>
<td>ODCOND&lt;sup&gt;(2)&lt;/sup&gt;</td>
<td>ODCD7</td>
<td>ODCD6</td>
<td>ODCD5</td>
<td>ODCD4</td>
<td>ODCD3</td>
<td>ODCD2</td>
<td>ODCD1</td>
<td>ODCD0</td>
<td>268</td>
</tr>
<tr>
<td>3A71h</td>
<td>WPUDP&lt;sup&gt;(3)&lt;/sup&gt;</td>
<td>WPUD7</td>
<td>WPUD6</td>
<td>WPUD5</td>
<td>WPUD4</td>
<td>WPUD3</td>
<td>WPUD2</td>
<td>WPUD1</td>
<td>WPUD0</td>
<td>267</td>
</tr>
<tr>
<td>3A70h</td>
<td>ANSELD&lt;sup&gt;(4)&lt;/sup&gt;</td>
<td>ANSELD7</td>
<td>ANSELD6</td>
<td>ANSELD5</td>
<td>ANSELD4</td>
<td>ANSELD3</td>
<td>ANSELD2</td>
<td>ANSELD1</td>
<td>ANSELD0</td>
<td>266</td>
</tr>
<tr>
<td>3A6Fh-3A6Ch</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>unimplemented</td>
<td>266</td>
</tr>
<tr>
<td>3A68h</td>
<td>RC4I2C</td>
<td>—</td>
<td>SLEW</td>
<td>PU</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>TH</td>
<td>263</td>
</tr>
<tr>
<td>3A6Ah</td>
<td>RC3I2C</td>
<td>—</td>
<td>SLEW</td>
<td>PU</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>TH</td>
<td>263</td>
</tr>
<tr>
<td>3A69h</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>reserved, maintain as '0'</td>
<td>266</td>
</tr>
<tr>
<td>3A68h</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>reserved, maintain as '0'</td>
<td>266</td>
</tr>
<tr>
<td>3A67h</td>
<td>IOCCF</td>
<td>IOCCF7</td>
<td>IOCCF6</td>
<td>IOCCF5</td>
<td>IOCCF4</td>
<td>IOCCF3</td>
<td>IOCCF2</td>
<td>IOCCF1</td>
<td>IOCCF0</td>
<td>287</td>
</tr>
<tr>
<td>3A66h</td>
<td>IOCCN</td>
<td>IOCCN7</td>
<td>IOCCN6</td>
<td>IOCCN5</td>
<td>IOCCN4</td>
<td>IOCCN3</td>
<td>IOCCN2</td>
<td>IOCCN1</td>
<td>IOCCN0</td>
<td>287</td>
</tr>
<tr>
<td>3A65h</td>
<td>IOCCP</td>
<td>IOCCP7</td>
<td>IOCCP6</td>
<td>IOCCP5</td>
<td>IOCCP4</td>
<td>IOCCP3</td>
<td>IOCCP2</td>
<td>IOCCP1</td>
<td>IOCCP0</td>
<td>287</td>
</tr>
<tr>
<td>3A64h</td>
<td>INLVLC</td>
<td>INLVLC7</td>
<td>INLVLC6</td>
<td>INLVLC5</td>
<td>INLVLC4</td>
<td>INLVLC3</td>
<td>INLVLC2</td>
<td>INLVLC1</td>
<td>INLVLC0</td>
<td>270</td>
</tr>
<tr>
<td>3A63h</td>
<td>SRRCONC</td>
<td>SLRC7</td>
<td>SLRC6</td>
<td>SLRC5</td>
<td>SLRC4</td>
<td>SLRC3</td>
<td>SLRC2</td>
<td>SLRC1</td>
<td>SLRC0</td>
<td>269</td>
</tr>
<tr>
<td>3A62h</td>
<td>ODCONC</td>
<td>ODCC7</td>
<td>ODCC6</td>
<td>ODCC5</td>
<td>ODCC4</td>
<td>ODCC3</td>
<td>ODCC2</td>
<td>ODCC1</td>
<td>ODCC0</td>
<td>268</td>
</tr>
<tr>
<td>3A61h</td>
<td>WPUC</td>
<td>WPUC7</td>
<td>WPUC6</td>
<td>WPUC5</td>
<td>WPUC4</td>
<td>WPUC3</td>
<td>WPUC2</td>
<td>WPUC1</td>
<td>WPUC0</td>
<td>267</td>
</tr>
<tr>
<td>3A60h</td>
<td>ANSELC</td>
<td>ANSELC7</td>
<td>ANSELC6</td>
<td>ANSELC5</td>
<td>ANSELC4</td>
<td>ANSELC3</td>
<td>ANSELC2</td>
<td>ANSELC1</td>
<td>ANSELC0</td>
<td>266</td>
</tr>
<tr>
<td>3A5Fh-3A5Ch</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>unimplemented</td>
<td>266</td>
</tr>
</tbody>
</table>

**Legend:**

- x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

**Note:**

1: Unimplemented in LF devices.
2: Unimplemented in PIC18(L)F26/27K42.
4: Unimplemented in PIC18(L)F45/55K42.
### TABLE 42-1: REGISTER FILE SUMMARY FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3A58h</td>
<td>RB2I2C</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>TH</td>
</tr>
<tr>
<td>3A5Ah</td>
<td>RB1I2C</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>TH</td>
</tr>
<tr>
<td>3A5h</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>Reserved, maintain as ‘0’</td>
</tr>
<tr>
<td>3A58h</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>Reserved, maintain as ‘0’</td>
</tr>
<tr>
<td>3A57h</td>
<td>IOCBF</td>
<td>IOCBF7</td>
<td>IOCBF6</td>
<td>IOCBF5</td>
<td>IOCBF4</td>
<td>IOCBF3</td>
<td>IOCBF2</td>
<td>IOCBF1</td>
<td>IOCBF0</td>
<td>287</td>
</tr>
<tr>
<td>3A56h</td>
<td>IOCBN</td>
<td>IOCBN7</td>
<td>IOCBN6</td>
<td>IOCBN5</td>
<td>IOCBN4</td>
<td>IOCBN3</td>
<td>IOCBN2</td>
<td>IOCBN1</td>
<td>IOCBN0</td>
<td>287</td>
</tr>
<tr>
<td>3A55h</td>
<td>IOCBP</td>
<td>IOCBP7</td>
<td>IOCBP6</td>
<td>IOCBP5</td>
<td>IOCBP4</td>
<td>IOCBP3</td>
<td>IOCBP2</td>
<td>IOCBP1</td>
<td>IOCBP0</td>
<td>287</td>
</tr>
<tr>
<td>3A54h</td>
<td>INLVLB</td>
<td>INLVLB7</td>
<td>INLVLB6</td>
<td>INLVLB5</td>
<td>INLVLB4</td>
<td>INLVLB3</td>
<td>INLVLB2</td>
<td>INLVLB1</td>
<td>INLVLB0</td>
<td>270</td>
</tr>
<tr>
<td>3A53h</td>
<td>SLRCONB</td>
<td>SLRBD7</td>
<td>SLRBD6</td>
<td>SLRBD5</td>
<td>SLRBD4</td>
<td>SLRBD3</td>
<td>SLRBD2</td>
<td>SLRBD1</td>
<td>SLRBD0</td>
<td>269</td>
</tr>
<tr>
<td>3A52h</td>
<td>ODCONB</td>
<td>ODCB7</td>
<td>ODCB6</td>
<td>ODCB5</td>
<td>ODCB4</td>
<td>ODCB3</td>
<td>ODCB2</td>
<td>ODCB1</td>
<td>ODCB0</td>
<td>268</td>
</tr>
<tr>
<td>3A51h</td>
<td>WPUB</td>
<td>WPUB7</td>
<td>WPUB6</td>
<td>WPUB5</td>
<td>WPUB4</td>
<td>WPUB3</td>
<td>WPUB2</td>
<td>WPUB1</td>
<td>WPUB0</td>
<td>267</td>
</tr>
<tr>
<td>3A50h</td>
<td>ANSELB</td>
<td>ANSELB7</td>
<td>ANSELB6</td>
<td>ANSELB5</td>
<td>ANSELB4</td>
<td>ANSELB3</td>
<td>ANSELB2</td>
<td>ANSELB1</td>
<td>ANSELB0</td>
<td>266</td>
</tr>
<tr>
<td>3A4Fh - 3A4Ah</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unimplemented</td>
</tr>
<tr>
<td>3A49h</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reserved, maintain as ‘0’</td>
</tr>
<tr>
<td>3A48h</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reserved, maintain as ‘0’</td>
</tr>
<tr>
<td>3A47h</td>
<td>IOCAF</td>
<td>IOCAF7</td>
<td>IOCAF6</td>
<td>IOCAF5</td>
<td>IOCAF4</td>
<td>IOCAF3</td>
<td>IOCAF2</td>
<td>IOCAF1</td>
<td>IOCAF0</td>
<td>287</td>
</tr>
<tr>
<td>3A46h</td>
<td>IOCAN</td>
<td>IOCAN7</td>
<td>IOCAN6</td>
<td>IOCAN5</td>
<td>IOCAN4</td>
<td>IOCAN3</td>
<td>IOCAN2</td>
<td>IOCAN1</td>
<td>IOCAN0</td>
<td>287</td>
</tr>
<tr>
<td>3A45h</td>
<td>IOCAP</td>
<td>IOCAP7</td>
<td>IOCAP6</td>
<td>IOCAP5</td>
<td>IOCAP4</td>
<td>IOCAP3</td>
<td>IOCAP2</td>
<td>IOCAP1</td>
<td>IOCAP0</td>
<td>287</td>
</tr>
<tr>
<td>3A44h</td>
<td>INLVLAA</td>
<td>INLVLAA7</td>
<td>INLVLAA6</td>
<td>INLVLAA5</td>
<td>INLVLAA4</td>
<td>INLVLAA3</td>
<td>INLVLAA2</td>
<td>INLVLAA1</td>
<td>INLVLAA0</td>
<td>270</td>
</tr>
<tr>
<td>3A43h</td>
<td>SLRCONAA</td>
<td>SLRA7</td>
<td>SLRA6</td>
<td>SLRA5</td>
<td>SLRA4</td>
<td>SLRA3</td>
<td>SLRA2</td>
<td>SLRA1</td>
<td>SLRA0</td>
<td>269</td>
</tr>
<tr>
<td>3A42h</td>
<td>ODCONA</td>
<td>ODCA7</td>
<td>ODCA6</td>
<td>ODCA5</td>
<td>ODCA4</td>
<td>ODCA3</td>
<td>ODCA2</td>
<td>ODCA1</td>
<td>ODCA0</td>
<td>268</td>
</tr>
<tr>
<td>3A41h</td>
<td>WPUA</td>
<td>WPUA7</td>
<td>WPUA6</td>
<td>WPUA5</td>
<td>WPUA4</td>
<td>WPUA3</td>
<td>WPUA2</td>
<td>WPUA1</td>
<td>WPUA0</td>
<td>267</td>
</tr>
<tr>
<td>3A40h</td>
<td>ANSELA</td>
<td>ANSELA7</td>
<td>ANSELA6</td>
<td>ANSELA5</td>
<td>ANSELA4</td>
<td>ANSELA3</td>
<td>ANSELA2</td>
<td>ANSELA1</td>
<td>ANSELA0</td>
<td>266</td>
</tr>
<tr>
<td>3A3Fh - 3A30h</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unimplemented</td>
</tr>
<tr>
<td>3A2Fh</td>
<td>RF7PPS2</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>280</td>
</tr>
<tr>
<td>3A2 Eh</td>
<td>RF6PPS2</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>280</td>
</tr>
<tr>
<td>3A20h</td>
<td>RF5PPS2</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>280</td>
</tr>
<tr>
<td>3A2Ch</td>
<td>RF4PPS2</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>280</td>
</tr>
<tr>
<td>3A28h</td>
<td>RF3PPS2</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>280</td>
</tr>
<tr>
<td>3A2Ah</td>
<td>RF2PPS2</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>280</td>
</tr>
<tr>
<td>3A29h</td>
<td>RF1PPS2</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>280</td>
</tr>
<tr>
<td>3A28h</td>
<td>RF0PPS2</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>280</td>
</tr>
<tr>
<td>3A27h - 3A23h</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unimplemented</td>
</tr>
<tr>
<td>3A22h</td>
<td>RE2PPS2</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>280</td>
</tr>
<tr>
<td>3A21h</td>
<td>RE1PPS2</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>280</td>
</tr>
<tr>
<td>3A20h</td>
<td>RE0PPS2</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>280</td>
</tr>
<tr>
<td>3A1Fh</td>
<td>RD7PPS2</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>280</td>
</tr>
<tr>
<td>3A1Ch</td>
<td>RD6PPS2</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>280</td>
</tr>
<tr>
<td>3A1Dh</td>
<td>RD5PPS2</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>280</td>
</tr>
<tr>
<td>3A1Ch</td>
<td>RD4PPS2</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>280</td>
</tr>
<tr>
<td>3A1Bh</td>
<td>RD3PPS2</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>280</td>
</tr>
<tr>
<td>3A1Ah</td>
<td>RD2PPS2</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>280</td>
</tr>
<tr>
<td>3A19h</td>
<td>RD1PPS2</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>280</td>
</tr>
<tr>
<td>3A18h</td>
<td>RD0PPS2</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>280</td>
</tr>
<tr>
<td>3A17h</td>
<td>RC7PPS2</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>280</td>
</tr>
</tbody>
</table>

**Legend:**
- x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

**Note:**
1: Unimplemented in LF devices.
2: Unimplemented in PIC18(L)F26/27K42.
4: Unimplemented in PIC18(L)F45/55K42.
TABLE 42-1: REGISTER FILE SUMMARY FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3A16h</td>
<td>RC6PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>RC6PPS0</td>
</tr>
<tr>
<td>3A15h</td>
<td>RC5PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>RC5PPS0</td>
</tr>
<tr>
<td>3A14h</td>
<td>RC4PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>RC4PPS0</td>
</tr>
<tr>
<td>3A13h</td>
<td>RC3PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>RC3PPS0</td>
</tr>
<tr>
<td>3A12h</td>
<td>RC2PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>RC2PPS0</td>
</tr>
<tr>
<td>3A11h</td>
<td>RC1PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>RC1PPS0</td>
</tr>
<tr>
<td>3A10h</td>
<td>RC0PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>RC0PPS0</td>
</tr>
<tr>
<td>3A0Fh</td>
<td>RB7PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>RB7PPS0</td>
</tr>
<tr>
<td>3A0Eh</td>
<td>RB6PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>RB6PPS0</td>
</tr>
<tr>
<td>3A0Dh</td>
<td>RB5PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>RB5PPS0</td>
</tr>
<tr>
<td>3A0Ch</td>
<td>RB4PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>RB4PPS0</td>
</tr>
<tr>
<td>3A0Bh</td>
<td>RB3PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>RB3PPS0</td>
</tr>
<tr>
<td>3A0Ah</td>
<td>RB2PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>RB2PPS0</td>
</tr>
<tr>
<td>3A09h</td>
<td>RB1PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>RB1PPS0</td>
</tr>
<tr>
<td>3A08h</td>
<td>RB0PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>RB0PPS0</td>
</tr>
<tr>
<td>3A07h</td>
<td>RA7PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>RA7PPS0</td>
</tr>
<tr>
<td>3A06h</td>
<td>RA6PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>RA6PPS0</td>
</tr>
<tr>
<td>3A05h</td>
<td>RA5PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>RA5PPS0</td>
</tr>
<tr>
<td>3A04h</td>
<td>RA4PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>RA4PPS0</td>
</tr>
<tr>
<td>3A03h</td>
<td>RA3PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>RA3PPS0</td>
</tr>
<tr>
<td>3A02h</td>
<td>RA2PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>RA2PPS0</td>
</tr>
<tr>
<td>3A01h</td>
<td>RA1PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>RA1PPS0</td>
</tr>
<tr>
<td>3A00h</td>
<td>RA0PPS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>RA0PPS0</td>
</tr>
<tr>
<td>39FFh</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>39F7h</td>
<td>SCANPR</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>PR</td>
</tr>
<tr>
<td>39F6h</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>39F4h</td>
<td>DMA2PR</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>PR</td>
</tr>
<tr>
<td>39F3h</td>
<td>DMA1PR</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>PR</td>
</tr>
<tr>
<td>39F2h</td>
<td>MAINPR</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>PR</td>
</tr>
<tr>
<td>39F1h</td>
<td>ISRPR</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>PR</td>
</tr>
<tr>
<td>39F0h</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>39Eh</td>
<td>PRLOCK</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>PRLOCKED</td>
</tr>
<tr>
<td>39Eh</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>39Eh</td>
<td>NVMCON2</td>
<td>NVMCON2</td>
<td></td>
<td>211</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>39Eh</td>
<td>NVMCON1</td>
<td>REG</td>
<td>FREE</td>
<td>WRERR</td>
<td>WREN</td>
<td>WR</td>
<td>RD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>39E4h</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>39E3h</td>
<td>NVMDAT</td>
<td>DAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>39E2h</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>39E1h</td>
<td>NVMADRh(4)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ADR</td>
</tr>
<tr>
<td>39E0h</td>
<td>NVMADR</td>
<td>ADR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>39Dh</td>
<td>OSCFRQ</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>FRQ</td>
</tr>
<tr>
<td>39Dh</td>
<td>OSC TUNE</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>TUN</td>
</tr>
<tr>
<td>39Dh</td>
<td>OSCEN</td>
<td>EXTEN</td>
<td>HFOEN</td>
<td>MFOEN</td>
<td>LFOEN</td>
<td>SOSEN</td>
<td>ADOEN</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>39Dh</td>
<td>OSCSTAT</td>
<td>EXTOR</td>
<td>HFOR</td>
<td>MFOR</td>
<td>LFOR</td>
<td>SOR</td>
<td>ADOR</td>
<td>—</td>
<td>—</td>
<td>PLLR</td>
</tr>
<tr>
<td>39Dh</td>
<td>OSCCON3</td>
<td>CSWHOLD</td>
<td>SOSCPWR</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
</tbody>
</table>

Legend: x = unknown, u = unchanged, — = unimplemented, q = value depends on condition
Note 1: Unimplemented in LF devices.
2: Unimplemented in PIC18(L)F26/27K42.
4: Unimplemented in PIC18(L)F45/55K42.
### TABLE 42-1: REGISTER FILE SUMMARY FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on page</th>
</tr>
</thead>
<tbody>
<tr>
<td>39D0h</td>
<td>OSCCON1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CDIV</td>
</tr>
<tr>
<td>39D1h</td>
<td>VREGPM</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>39D2h</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>39D3h</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>39D4h</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>39D5h</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>39D6h</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>39D7h</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>39D8h</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>39D9h</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>39DAh</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>39DBh</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>39DCh</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>39DDh</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>39DEh</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>39DFh</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
</tr>
</tbody>
</table>

**Legend:**
- x = unknown
- u = unchanged
- — = unimplemented
- q = value depends on condition

**Note:**
1. Unimplemented in LF devices.
2. Unimplemented in PIC18(L)F26/27K42.
4. Unimplemented in PIC18(L)F45/55K42.
### TABLE 42-1: REGISTER FILE SUMMARY FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3989h</td>
<td>IPR9</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CLC3IP</td>
<td>165</td>
</tr>
<tr>
<td>3988h</td>
<td>IPR8</td>
<td>TMR5GIP</td>
<td>TMR5IP</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>164</td>
</tr>
<tr>
<td>3987h</td>
<td>IPR7</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>INT2IP</td>
<td>CLC2IP</td>
<td>CWG2IP</td>
<td>—</td>
<td>—</td>
<td>164</td>
</tr>
<tr>
<td>3986h</td>
<td>IPR6</td>
<td>TMR3GIP</td>
<td>TMR3IP</td>
<td>U2IP</td>
<td>U2EIP</td>
<td>U2TXIP</td>
<td>U2RXIP</td>
<td>—</td>
<td>CCP2IP</td>
<td>163</td>
</tr>
<tr>
<td>3985h</td>
<td>IPR5</td>
<td>I2C2TXIP</td>
<td>I2C2RXIP</td>
<td>DMA2AIIP</td>
<td>DMA2ORIP</td>
<td>DMA2DCN-</td>
<td>DMA2SCN-</td>
<td>C2IP</td>
<td>TMR6IP</td>
<td>162</td>
</tr>
<tr>
<td>3984h</td>
<td>IPR4</td>
<td>CLC1IP</td>
<td>CWG1IP</td>
<td>NCO1IP</td>
<td>—</td>
<td>CCP1IP</td>
<td>TMR2IP</td>
<td>TMR1GIP</td>
<td>TMR1IP</td>
<td>161</td>
</tr>
<tr>
<td>3983h</td>
<td>IPR3</td>
<td>TMR0IP</td>
<td>U1IP</td>
<td>U1EIP</td>
<td>U1TXIP</td>
<td>U1RXIP</td>
<td>U2C1EIP</td>
<td>U2C1IP</td>
<td>U2C1TXIP</td>
<td>160</td>
</tr>
<tr>
<td>3982h</td>
<td>IPR2</td>
<td>I2C1RXIP</td>
<td>SPI1IP</td>
<td>SPI1TXIP</td>
<td>SPI1RXIP</td>
<td>DMA1AIIP</td>
<td>DMA1ORIP</td>
<td>DMA1DCN-</td>
<td>DMA1SCN-</td>
<td>159</td>
</tr>
<tr>
<td>3981h</td>
<td>IPR1</td>
<td>SM1TPWAPI</td>
<td>SM1TPRAIP</td>
<td>SM1TI</td>
<td>C1IP</td>
<td>ADTIP</td>
<td>ADIP</td>
<td>ZCIDIP</td>
<td>INT0IP</td>
<td>158</td>
</tr>
<tr>
<td>3980h</td>
<td>IPR0</td>
<td>IOCIP</td>
<td>CRCIP</td>
<td>SCAN1P</td>
<td>NV1P</td>
<td>CSW1P</td>
<td>OS1P</td>
<td>HLV1IP</td>
<td>SW1P</td>
<td>157</td>
</tr>
<tr>
<td>397Fh - 397eh</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>226</td>
</tr>
<tr>
<td>397Dh</td>
<td>SCANTRIG</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>225</td>
</tr>
<tr>
<td>397Ch</td>
<td>SCANCON0</td>
<td>EN</td>
<td>TRIGEN</td>
<td>SGO</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>222</td>
</tr>
<tr>
<td>397Bh</td>
<td>SCANHADRU</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>224</td>
</tr>
<tr>
<td>397Ah</td>
<td>SCANHADRH</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>225</td>
</tr>
<tr>
<td>3979h</td>
<td>SCANHADRL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>225</td>
</tr>
<tr>
<td>3978h</td>
<td>SCANLADRU</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>223</td>
</tr>
<tr>
<td>3977h</td>
<td>SCANLADRH</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>223</td>
</tr>
<tr>
<td>3976h</td>
<td>SCANLADRL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>224</td>
</tr>
<tr>
<td>3975h - 396Ah</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>218</td>
</tr>
<tr>
<td>3969h</td>
<td>CRCCON1</td>
<td>DLEN</td>
<td>PLEN</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>218</td>
</tr>
<tr>
<td>3968h</td>
<td>CRCCON0</td>
<td>EN</td>
<td>CRCGO</td>
<td>BUSY</td>
<td>ACCM</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>221</td>
</tr>
<tr>
<td>3967h</td>
<td>CRXORH</td>
<td>X15</td>
<td>X14</td>
<td>X13</td>
<td>X12</td>
<td>X11</td>
<td>X10</td>
<td>X9</td>
<td>X8</td>
<td>220</td>
</tr>
<tr>
<td>3966h</td>
<td>CRXORL</td>
<td>X7</td>
<td>X6</td>
<td>X5</td>
<td>X4</td>
<td>X3</td>
<td>X2</td>
<td>X1</td>
<td>—</td>
<td>221</td>
</tr>
<tr>
<td>3965h</td>
<td>CRCSHFTL</td>
<td>SHFT15</td>
<td>SHFT14</td>
<td>SHFT13</td>
<td>SHFT12</td>
<td>SHFT11</td>
<td>SHFT10</td>
<td>SHFT9</td>
<td>SHFT8</td>
<td>220</td>
</tr>
<tr>
<td>3964h</td>
<td>CRCSHFTL</td>
<td>SHFT7</td>
<td>SHFT6</td>
<td>SHFT5</td>
<td>SHFT4</td>
<td>SHFT3</td>
<td>SHFT2</td>
<td>SHFT1</td>
<td>SHFT0</td>
<td>220</td>
</tr>
<tr>
<td>3963h</td>
<td>CRCACCH</td>
<td>ACC15</td>
<td>ACC14</td>
<td>ACC13</td>
<td>ACC12</td>
<td>ACC11</td>
<td>ACC10</td>
<td>ACC9</td>
<td>ACC8</td>
<td>219</td>
</tr>
<tr>
<td>3962h</td>
<td>CRCACCL</td>
<td>ACC7</td>
<td>ACC6</td>
<td>ACC5</td>
<td>ACC4</td>
<td>ACC3</td>
<td>ACC2</td>
<td>ACC1</td>
<td>ACC0</td>
<td>220</td>
</tr>
<tr>
<td>3961h</td>
<td>CRCDATL</td>
<td>DATA15</td>
<td>DATA14</td>
<td>DATA13</td>
<td>DATA12</td>
<td>DATA11</td>
<td>DATA10</td>
<td>DATA9</td>
<td>DATA8</td>
<td>219</td>
</tr>
<tr>
<td>3960h</td>
<td>CRCDATL</td>
<td>DATA7</td>
<td>DATA6</td>
<td>DATA5</td>
<td>DATA4</td>
<td>DATA3</td>
<td>DATA2</td>
<td>DATA1</td>
<td>DATA0</td>
<td>219</td>
</tr>
<tr>
<td>395Fh</td>
<td>WDTTMR</td>
<td>WDTTMR</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>185</td>
</tr>
<tr>
<td>395Eh</td>
<td>WDP5SH</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>184</td>
</tr>
<tr>
<td>395Dh</td>
<td>WDTPLS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>184</td>
</tr>
<tr>
<td>395Ch</td>
<td>WDTCN1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>183</td>
</tr>
<tr>
<td>3958h</td>
<td>WDTCN0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>182</td>
</tr>
<tr>
<td>395Ah - 394Dh</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>167</td>
</tr>
<tr>
<td>389Fh</td>
<td>IVTADU</td>
<td>AD</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>167</td>
</tr>
<tr>
<td>389Eh</td>
<td>IVTADH</td>
<td>AD</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>167</td>
</tr>
<tr>
<td>389Dh</td>
<td>IVTADL</td>
<td>AD</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>167</td>
</tr>
<tr>
<td>389Ch - 3891h</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>125</td>
</tr>
<tr>
<td>3890h</td>
<td>PRODH_SHAD</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>125</td>
</tr>
<tr>
<td>388Fh</td>
<td>PRODL_SHAD</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>125</td>
</tr>
<tr>
<td>388Eh</td>
<td>FSR2H_SHAD</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>125</td>
</tr>
</tbody>
</table>

**Legend:**  
x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

**Note:**  
1: Unimplemented in LF devices.  
2: Unimplemented in PIC18(L)F26/27K42.  
4: Unimplemented in PIC18(L)F45/55K42.
# TABLE 42-1: REGISTER FILE SUMMARY FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3880h</td>
<td>FSR2L_SHAD</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>FSR2L</td>
</tr>
<tr>
<td>3881h</td>
<td>FSR1H_SHAD</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>FSR1H</td>
</tr>
<tr>
<td>3882h</td>
<td>FSR1L_SHAD</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>FSR1L</td>
</tr>
<tr>
<td>3883h</td>
<td>FSR0H_SHAD</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>FSR0H</td>
</tr>
<tr>
<td>3884h</td>
<td>FSR0L_SHAD</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>FSR0L</td>
</tr>
<tr>
<td>3885h</td>
<td>PCLATU_SHAD</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>PCU</td>
</tr>
<tr>
<td>3886h</td>
<td>PCLATH_SHAD</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>PCH</td>
</tr>
<tr>
<td>3887h</td>
<td>BSR_SHAD</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>BSR</td>
</tr>
<tr>
<td>3888h</td>
<td>WREG_SHAD</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>WREG</td>
</tr>
<tr>
<td>3889h</td>
<td>STATUS_SHAD</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>STATUS</td>
</tr>
<tr>
<td>388Ah</td>
<td>SHADCON</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>SHADLO</td>
</tr>
<tr>
<td>388Bh</td>
<td>BSR_CSHAD</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>BSR</td>
</tr>
<tr>
<td>388Ch</td>
<td>WREG_CSHAD</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>WREG</td>
</tr>
<tr>
<td>388Dh</td>
<td>STATUS_CSHAD</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>STATUS_CSHAD</td>
</tr>
</tbody>
</table>

**Legend:**
- x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

**Note:**
1: Unimplemented in LF devices.
2: Unimplemented in PIC18(L)F26/27K42.
4: Unimplemented in PIC18(L)F45/55K42.
43.0 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

43.1 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
43.2 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

43.3 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

43.4 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/library 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

43.5 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
43.6  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.

43.7  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.

43.8  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.

43.9  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™).

43.10 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.
43.11 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, KEELog® 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) for the complete list of demonstration, development and evaluation kits.

43.12 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®
44.0 ELECTRICAL SPECIFICATIONS

44.1 Absolute Maximum Ratings(†)

Ambient temperature under bias ........................................................................................................... -40°C to +125°C
Storage temperature ........................................................................................................................ -65°C to +150°C

Voltage on pins with respect to Vss

- on VDD pin
  - PIC18F26/45/46/55/56K42 ............................................................................................... -0.3V to +6.5V
  - PIC18(L)F26/27/45/46/47/55/56/57K42 ........................................................................ -0.3V to +4.0V
- on MCLR pin .......................................................................................................................... . -0.3V to +9.0V
- on all other pins ............................................................................................................ -0.3V to (VDD + 0.3V)

Maximum current

- on Vss pin(1)
  - -40°C ≤ TA ≤ +85°C .............................................................................................................. 350 mA
  - 85°C < TA ≤ +125°C ........................................................................................................... 120 mA
- on VDD pin for 28-Pin devices(1)
  - -40°C ≤ TA ≤ +85°C .............................................................................................................. 250 mA
  - 85°C < TA ≤ +125°C ............................................................................................................. 85 mA
- on VDD pin for 40-Pin devices(1)
  - -40°C ≤ TA ≤ +85°C .............................................................................................................. 350 mA
  - 85°C < TA ≤ +125°C ............................................................................................................. 120 mA
- on any standard I/O pin ...................................................................................................................... ±50 mA

Clamp current, Ik (VPIN < 0 or VPIN > VDD) ................................................................................................... ±20 mA

Total power dissipation(2) ....................................................................................................................... 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 44-8 to calculate device
specifications.

2: Power dissipation is calculated as follows:

\[
P_{\text{DIS}} = V_{\text{DD}} \times (I_{\text{DD}} - \sum I_{\text{OH}}) + \sum \left( (V_{\text{DD}} - V_{\text{OH}}) \times I_{\text{OH}} \right) + \sum (V_{\text{OI}} \times I_{\text{OL}})
\]

† 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.
44.2 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: \( TA_{MIN} \leq TA \leq TA_{MAX} \)

**VDD — Operating Supply Voltage\(^{(1)}\)**

<table>
<thead>
<tr>
<th>Device</th>
<th>VDDMIN (Fosc ≤ 16 MHz)</th>
<th>VDDMIN (Fosc ≤ 32 MHz)</th>
<th>VDDMIN (Fosc ≤ 64 MHz)</th>
<th>VDDMAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIC18(L)F26/27/45/46/47/55/56/57K42</td>
<td>+1.8V</td>
<td>+2.5V</td>
<td>+2.7V</td>
<td>+3.6V</td>
</tr>
<tr>
<td>PIC18F26/45/46/55/56K42</td>
<td>+2.3V</td>
<td>+2.5V</td>
<td>+3.0V</td>
<td>+5.5V</td>
</tr>
</tbody>
</table>

**TA — Operating Ambient Temperature Range**

<table>
<thead>
<tr>
<th>Temperature Type</th>
<th>TA_MIN</th>
<th>TA_MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Industrial</td>
<td>-40°C</td>
<td>+85°C</td>
</tr>
<tr>
<td>Extended</td>
<td>-40°C</td>
<td>+125°C</td>
</tr>
</tbody>
</table>

**Note 1:** See Parameter Supply Voltage, DS Characteristics: Supply Voltage.
FIGURE 44-1: VOLTAGE FREQUENCY GRAPH, -40°C ≤ TA ≤ +125°C, PIC18F26/45/46/55/56K42 ONLY

Note 1: The shaded region indicates the permissible combinations of voltage and frequency.
2: Refer to Table 44-9 for each Oscillator mode's supported frequencies.


Note 1: The shaded region indicates the permissible combinations of voltage and frequency.
2: Refer to Table 44-9 for each Oscillator mode's supported frequencies.
## 44.3 DC Characteristics

### TABLE 44-1: SUPPLY VOLTAGE

<table>
<thead>
<tr>
<th>Param. No.</th>
<th>Sym.</th>
<th>Characteristic</th>
<th>Min.</th>
<th>Typ.†</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>D002</td>
<td>VDD</td>
<td>Supply Voltage</td>
<td>1.8</td>
<td>3.6</td>
<td>2.5</td>
<td>V</td>
<td>FOSC ≤ 16 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2.7</td>
<td>3.6</td>
<td>2.7</td>
<td>V</td>
<td>FOSC &gt; 32 MHz</td>
</tr>
<tr>
<td>D002</td>
<td>VDD</td>
<td></td>
<td>2.3</td>
<td>5.5</td>
<td>2.5</td>
<td>V</td>
<td>FOSC ≤ 16 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2.7</td>
<td>5.5</td>
<td>2.7</td>
<td>V</td>
<td>FOSC &gt; 32 MHz</td>
</tr>
<tr>
<td>D002</td>
<td>VDD</td>
<td></td>
<td>2.7</td>
<td>5.5</td>
<td>2.7</td>
<td>V</td>
<td>FOSC &gt; 64 MHz</td>
</tr>
<tr>
<td>D003</td>
<td>VDR</td>
<td>RAM Data Retention(1)</td>
<td>1.5</td>
<td>—</td>
<td>—</td>
<td>V</td>
<td>Device in Sleep mode</td>
</tr>
<tr>
<td>D003</td>
<td>VDR</td>
<td></td>
<td>1.7</td>
<td>—</td>
<td>—</td>
<td>V</td>
<td>Device in Sleep mode</td>
</tr>
<tr>
<td>D004</td>
<td>VPOR</td>
<td>Power-on Reset Release Voltage(2)</td>
<td>—</td>
<td>1.6</td>
<td>—</td>
<td>V</td>
<td>BOR or LPBOR disabled(3)</td>
</tr>
<tr>
<td>D004</td>
<td>VPOR</td>
<td></td>
<td>—</td>
<td>1.6</td>
<td>—</td>
<td>V</td>
<td>BOR or LPBOR disabled(3)</td>
</tr>
<tr>
<td>D005</td>
<td>VPORR</td>
<td>Power-on Reset Rearm Voltage(2)</td>
<td>—</td>
<td>0.8</td>
<td>—</td>
<td>V</td>
<td>BOR or LPBOR disabled(3)</td>
</tr>
<tr>
<td>D005</td>
<td>VPORR</td>
<td></td>
<td>—</td>
<td>1.5</td>
<td>—</td>
<td>V</td>
<td>BOR or LPBOR disabled(3)</td>
</tr>
<tr>
<td>D006</td>
<td>SVDD</td>
<td>Vdd Rise Rate to ensure internal Power-on Reset signal(2)</td>
<td>0.05</td>
<td>—</td>
<td>—</td>
<td>V/ms</td>
<td>BOR or LPBOR disabled(3)</td>
</tr>
<tr>
<td>D006</td>
<td>SVDD</td>
<td></td>
<td>0.05</td>
<td>—</td>
<td>—</td>
<td>V/ms</td>
<td>BOR or LPBOR disabled(3)</td>
</tr>
</tbody>
</table>

† 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.

**Note 2:** See Figure 44-3, POR and POR REARM with Slow Rising Vdd.

**Note 3:** See Table 44-13 for BOR and LPBOR trip point information.
FIGURE 44-3: POR AND POR REARM WITH SLOW RISING VDD

Note 1: When NPOR is low, the device is held in Reset.
2: $\text{TPOR}$ 1 $\mu$s typical.
3: $\text{TVLOW}$ 2.7 $\mu$s typical.
### TABLE 44-2: SUPPLY CURRENT (I\(\text{DD}(1,2,4)\))

<table>
<thead>
<tr>
<th>Param. No.</th>
<th>Symbol</th>
<th>Device Characteristics</th>
<th>Min.</th>
<th>Typ.†</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>D100</td>
<td>(I_{\text{DD}XT4})</td>
<td>XT = 4 MHz</td>
<td>—</td>
<td>741</td>
<td>1000</td>
<td>(\mu\text{A})</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>D101</td>
<td>(I_{\text{DD}XT4})</td>
<td>XT = 4 MHz</td>
<td>—</td>
<td>770</td>
<td>1200</td>
<td>(\mu\text{A})</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>D100A</td>
<td>(I_{\text{DD}XT4})</td>
<td>XT = 4 MHz</td>
<td>—</td>
<td>390</td>
<td>—</td>
<td>(\mu\text{A})</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>D100A</td>
<td>(I_{\text{DD}XT4})</td>
<td>XT = 4 MHz</td>
<td>—</td>
<td>620</td>
<td>—</td>
<td>(\mu\text{A})</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>D101</td>
<td>(I_{\text{DDHFO16}})</td>
<td>HFINTOSC = 16 MHz</td>
<td>—</td>
<td>2.3</td>
<td>4</td>
<td>mA</td>
<td>3.0V</td>
<td>—</td>
</tr>
<tr>
<td>D102</td>
<td>(I_{\text{DDHFOPLL}})</td>
<td>HFINTOSC = 64 MHz</td>
<td>—</td>
<td>8.4</td>
<td>15</td>
<td>mA</td>
<td>3.0V</td>
<td>—</td>
</tr>
<tr>
<td>D102A</td>
<td>(I_{\text{DDHFOPLL}})</td>
<td>HFINTOSC = 64 MHz</td>
<td>—</td>
<td>5</td>
<td>—</td>
<td>mA</td>
<td>3.0V</td>
<td>—</td>
</tr>
<tr>
<td>D103</td>
<td>(I_{\text{DDHPSPLL64}})</td>
<td>HS+PLL = 64 MHz</td>
<td>—</td>
<td>8.4</td>
<td>15</td>
<td>mA</td>
<td>3.0V</td>
<td>—</td>
</tr>
<tr>
<td>D104</td>
<td>(I_{\text{IDLEDLE}})</td>
<td>IDLE mode, HFINTOSC = 16 MHz</td>
<td>—</td>
<td>1.9</td>
<td>—</td>
<td>mA</td>
<td>3.0V</td>
<td>—</td>
</tr>
<tr>
<td>D105</td>
<td>(I_{\text{IDDDOZE}})</td>
<td>DOZE mode, HFINTOSC = 16 MHz, Doze Ratio = 16</td>
<td>—</td>
<td>1.9</td>
<td>—</td>
<td>mA</td>
<td>3.0V</td>
<td>—</td>
</tr>
</tbody>
</table>

† 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 \(I_{\text{DD}}\) measurements in active operation mode are: OSC1 = external square wave, from rail-to-rail; all I/O pins are outputs driven low; MCLR = V\(\text{DD}\); 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. \(I_{\text{DDDOZE}} = [I_{\text{DDIDLE}}(N-1)/N] + I_{\text{DDHFO16}}/N\) where \(N = \text{DOZE Ratio (Register 10-2)}\).
4. PMD bits are all in the default state, no modules are disabled.
### TABLE 44-3: SUPPLY CURRENT (I_DD)\(^{(1,2,4)}\)

<table>
<thead>
<tr>
<th>Param. No.</th>
<th>Symbol</th>
<th>Device Characteristics</th>
<th>Min.</th>
<th>Typ.†</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>D100</td>
<td>I_DD_XT4</td>
<td>XT = 4 MHz</td>
<td>—</td>
<td>625</td>
<td>1200</td>
<td>μA</td>
<td>—</td>
</tr>
<tr>
<td>D100</td>
<td>I_DD_XT4</td>
<td>XT = 4 MHz</td>
<td>—</td>
<td>825</td>
<td>1400</td>
<td>μA</td>
<td>—</td>
</tr>
<tr>
<td>D100A</td>
<td>I_DD_XT4</td>
<td>XT = 4 MHz</td>
<td>—</td>
<td>425</td>
<td>—</td>
<td>μA</td>
<td>—</td>
</tr>
<tr>
<td>D100A</td>
<td>I_DD_XT4</td>
<td>XT = 4 MHz</td>
<td>—</td>
<td>665</td>
<td>—</td>
<td>μA</td>
<td>—</td>
</tr>
<tr>
<td>D101</td>
<td>I_DD_HF016</td>
<td>HFINTOSC = 16 MHz</td>
<td>—</td>
<td>2.9</td>
<td>5</td>
<td>mA</td>
<td>—</td>
</tr>
<tr>
<td>D101</td>
<td>I_DD_HF016</td>
<td>HFINTOSC = 16 MHz</td>
<td>—</td>
<td>3</td>
<td>5.1</td>
<td>mA</td>
<td>—</td>
</tr>
<tr>
<td>D101A</td>
<td>I_DD_HF016</td>
<td>HFINTOSC = 16 MHz</td>
<td>—</td>
<td>2</td>
<td>—</td>
<td>mA</td>
<td>—</td>
</tr>
<tr>
<td>D101A</td>
<td>I_DD_HF016</td>
<td>HFINTOSC = 16 MHz</td>
<td>—</td>
<td>2.1</td>
<td>—</td>
<td>mA</td>
<td>—</td>
</tr>
<tr>
<td>D102</td>
<td>I_DD_FOPLL</td>
<td>HFINTOSC = 64 MHz</td>
<td>—</td>
<td>10.7</td>
<td>17.5</td>
<td>mA</td>
<td>—</td>
</tr>
<tr>
<td>D102</td>
<td>I_DD_FOPLL</td>
<td>HFINTOSC = 64 MHz</td>
<td>—</td>
<td>11</td>
<td>18</td>
<td>mA</td>
<td>—</td>
</tr>
<tr>
<td>D102A</td>
<td>I_DD_FOPLL</td>
<td>HFINTOSC = 64 MHz</td>
<td>—</td>
<td>6.7</td>
<td>—</td>
<td>mA</td>
<td>—</td>
</tr>
<tr>
<td>D102A</td>
<td>I_DD_FOPLL</td>
<td>HFINTOSC = 64 MHz</td>
<td>—</td>
<td>6.9</td>
<td>—</td>
<td>mA</td>
<td>—</td>
</tr>
<tr>
<td>D103</td>
<td>I_DD_HSPLL64</td>
<td>HS+PLL = 64 MHz</td>
<td>—</td>
<td>10.7</td>
<td>17.5</td>
<td>mA</td>
<td>—</td>
</tr>
<tr>
<td>D103</td>
<td>I_DD_HSPLL64</td>
<td>HS+PLL = 64 MHz</td>
<td>—</td>
<td>11</td>
<td>18</td>
<td>mA</td>
<td>—</td>
</tr>
<tr>
<td>D103A</td>
<td>I_DD_HSPLL64</td>
<td>HS+PLL = 64 MHz</td>
<td>—</td>
<td>6.7</td>
<td>—</td>
<td>mA</td>
<td>—</td>
</tr>
<tr>
<td>D103A</td>
<td>I_DD_HSPLL64</td>
<td>HS+PLL = 64 MHz</td>
<td>—</td>
<td>6.9</td>
<td>—</td>
<td>mA</td>
<td>—</td>
</tr>
<tr>
<td>D104</td>
<td>I_DD_IDLE</td>
<td>IDLE mode, HFINTOSC = 16 MHz</td>
<td>—</td>
<td>2.0</td>
<td>—</td>
<td>mA</td>
<td>—</td>
</tr>
<tr>
<td>D104</td>
<td>I_DD_IDLE</td>
<td>IDLE mode, HFINTOSC = 16 MHz</td>
<td>—</td>
<td>2.1</td>
<td>—</td>
<td>mA</td>
<td>—</td>
</tr>
<tr>
<td>D105</td>
<td>I_DD_DOZE(^{(3)})</td>
<td>DOZE mode, HFINTOSC = 16 MHz, Doze Ratio = 16</td>
<td>—</td>
<td>2.0</td>
<td>—</td>
<td>mA</td>
<td>—</td>
</tr>
<tr>
<td>D105</td>
<td>I_DD_DOZE(^{(3)})</td>
<td>DOZE mode, HFINTOSC = 16 MHz, Doze Ratio = 16</td>
<td>—</td>
<td>2.1</td>
<td>—</td>
<td>mA</td>
<td>—</td>
</tr>
</tbody>
</table>

† 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 = VDD; 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*(N-1)/N] + I_DD\_HF016/N where N = DOZE Ratio (Register 10-2).
4. PMD bits are all in the default state, no modules are disabled.
### TABLE 44-4: POWER-DOWN CURRENT (IPD)\(^{(1,2)}\)

<table>
<thead>
<tr>
<th>PIC18F26/45/46/55/56K42</th>
<th>Standard Operating Conditions (unless otherwise stated)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>VDD Note</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Param. No.</th>
<th>Symbol</th>
<th>Device Characteristics</th>
<th>Min.</th>
<th>Typ.†</th>
<th>Max. (+85°C)</th>
<th>Max. (+125°C)</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>D200</td>
<td>IPD Base</td>
<td>—</td>
<td>0.07</td>
<td>2</td>
<td>6</td>
<td>3.0V</td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td>D200A</td>
<td>IPD Base</td>
<td>—</td>
<td>0.4</td>
<td>4</td>
<td>8</td>
<td>3.0V</td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td>D201</td>
<td>Low-Frequency Internal Oscillator/ WDT</td>
<td>—</td>
<td>0.9</td>
<td>3</td>
<td>7</td>
<td>3.0V</td>
<td>μA</td>
<td>VREGPM = 0</td>
</tr>
<tr>
<td>D201</td>
<td>Low-Frequency Internal Oscillator/ WDT</td>
<td>—</td>
<td>1.1</td>
<td>3</td>
<td>9</td>
<td>3.0V</td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td>D202</td>
<td>Secondary Oscillator (SoSsc)</td>
<td>—</td>
<td>0.75</td>
<td>5</td>
<td>9</td>
<td>3.0V</td>
<td>μA</td>
<td>LP mode</td>
</tr>
<tr>
<td>D202</td>
<td>Secondary Oscillator (SoSsc)</td>
<td>—</td>
<td>1.0</td>
<td>6.5</td>
<td>10</td>
<td>3.0V</td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td>D203</td>
<td>FVR</td>
<td>—</td>
<td>45</td>
<td>74</td>
<td>75</td>
<td>3.0V</td>
<td>μA</td>
<td>FVRCON = 0x81 or 0x84</td>
</tr>
<tr>
<td>D203</td>
<td>FVR</td>
<td>—</td>
<td>40</td>
<td>70</td>
<td>76</td>
<td>3.0V</td>
<td>μA</td>
<td>FVRCON = 0x81 or 0x84</td>
</tr>
<tr>
<td>D204</td>
<td>Brown-out Reset (BOR)</td>
<td>—</td>
<td>9.4</td>
<td>14</td>
<td>18</td>
<td>3.0V</td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td>D205</td>
<td>Low-Power Brown-out Reset (LPBOR)</td>
<td>—</td>
<td>0.2</td>
<td>3</td>
<td>6</td>
<td>3.0V</td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td>D206</td>
<td>High/Low Voltage Detect (HLVD)</td>
<td>—</td>
<td>9.5</td>
<td>14.8</td>
<td>18</td>
<td>3.0V</td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td>D207</td>
<td>ADC - Non-Converting</td>
<td>—</td>
<td>0.1</td>
<td>2</td>
<td>6</td>
<td>3.0V</td>
<td>μA</td>
<td>ADC not converting ((4))</td>
</tr>
<tr>
<td>D208</td>
<td>Comparator</td>
<td>—</td>
<td>43</td>
<td>49</td>
<td>50</td>
<td>3.0V</td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td>D208</td>
<td>Comparator</td>
<td>—</td>
<td>30</td>
<td>49</td>
<td>50</td>
<td>3.0V</td>
<td>μA</td>
<td></td>
</tr>
</tbody>
</table>

† 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 or IPD current and the additional current consumed when this peripheral is enabled. The peripheral 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 44-5: POWER-DOWN CURRENT (IPD)\(^{(1,2)}\)

<table>
<thead>
<tr>
<th>PIC18LF27/47/57K42</th>
<th>Standard Operating Conditions (unless otherwise stated)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PIC18F27/47/57K42</strong></td>
<td><strong>Standard Operating Conditions (unless otherwise stated)</strong></td>
</tr>
<tr>
<td><strong>Param. No.</strong></td>
<td><strong>Symbol</strong></td>
</tr>
<tr>
<td>D200</td>
<td>IPD</td>
</tr>
<tr>
<td>D200A</td>
<td>IPD</td>
</tr>
<tr>
<td>D201</td>
<td>IPD</td>
</tr>
<tr>
<td>D202</td>
<td>IPD</td>
</tr>
<tr>
<td>D203</td>
<td>IPD</td>
</tr>
<tr>
<td>D204</td>
<td>IPD</td>
</tr>
<tr>
<td>D205</td>
<td>IPD</td>
</tr>
<tr>
<td>D206</td>
<td>IPD</td>
</tr>
<tr>
<td>D207</td>
<td>IPD</td>
</tr>
<tr>
<td>D208</td>
<td>IPD</td>
</tr>
<tr>
<td>D209</td>
<td>IPD</td>
</tr>
</tbody>
</table>

\(\dagger\) 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 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 44-6: I/O PORTS

**Standard Operating Conditions** (unless otherwise stated)

<table>
<thead>
<tr>
<th>Param No.</th>
<th>Sym.</th>
<th>Characteristic</th>
<th>Min.</th>
<th>Typ†</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>VIL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D300</td>
<td>—</td>
<td>Input Low Voltage</td>
<td>—</td>
<td>—</td>
<td>0.8</td>
<td>V</td>
<td>4.5V ≤ VDD ≤ 5.5V</td>
</tr>
<tr>
<td>D301</td>
<td>—</td>
<td></td>
<td>—</td>
<td>—</td>
<td>0.15V</td>
<td>V</td>
<td>1.8V ≤ VDD &lt; 4.5V</td>
</tr>
<tr>
<td>D302</td>
<td>—</td>
<td></td>
<td>—</td>
<td>—</td>
<td>0.2V</td>
<td>V</td>
<td>2.0V ≤ VDD ≤ 5.5V</td>
</tr>
<tr>
<td>D303</td>
<td>—</td>
<td></td>
<td>—</td>
<td>—</td>
<td>0.3V</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>D304</td>
<td>—</td>
<td></td>
<td>—</td>
<td>—</td>
<td>0.8</td>
<td>V</td>
<td>2.7V ≤ VDD ≤ 5.5V</td>
</tr>
<tr>
<td>D305</td>
<td>—</td>
<td></td>
<td>—</td>
<td>—</td>
<td>0.8</td>
<td>V</td>
<td>1.8V ≤ VDD ≤ 5.5V</td>
</tr>
<tr>
<td>D306</td>
<td>—</td>
<td>MCLR</td>
<td>—</td>
<td>—</td>
<td>0.2V</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VIH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D320</td>
<td>2.0</td>
<td></td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>V</td>
<td>4.5V ≤ VDD ≤ 5.5V</td>
</tr>
<tr>
<td>D321</td>
<td>0.25VDD + 0.8</td>
<td></td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>V</td>
<td>1.8V ≤ VDD &lt; 4.5V</td>
</tr>
<tr>
<td>D322</td>
<td>0.8VDD</td>
<td></td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>V</td>
<td>2.0V ≤ VDD ≤ 5.5V</td>
</tr>
<tr>
<td>D323</td>
<td>0.7VDD</td>
<td></td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>D324</td>
<td>2.1</td>
<td></td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>V</td>
<td>2.7V ≤ VDD ≤ 5.5V</td>
</tr>
<tr>
<td>D325</td>
<td>1.35</td>
<td></td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>V</td>
<td>1.8V ≤ VDD ≤ 5.5V</td>
</tr>
<tr>
<td>D326</td>
<td>0.7VDD</td>
<td></td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IIL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D340</td>
<td>± 5</td>
<td></td>
<td>±125</td>
<td>—</td>
<td>—</td>
<td>nA</td>
<td>VSS ≤ VPIN ≤ VDD, Pin at high-impedance, 85°C</td>
</tr>
<tr>
<td>D341</td>
<td>± 5</td>
<td></td>
<td>±1000</td>
<td>—</td>
<td>—</td>
<td>nA</td>
<td>VSS ≤ VPIN ≤ VDD, Pin at high-impedance, 125°C</td>
</tr>
<tr>
<td>D342</td>
<td>± 50</td>
<td></td>
<td>±200</td>
<td>—</td>
<td>—</td>
<td>nA</td>
<td>VSS ≤ VPIN ≤ VDD, Pin at high-impedance, 85°C</td>
</tr>
<tr>
<td></td>
<td>IPUR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D350</td>
<td>25</td>
<td></td>
<td>120</td>
<td>200</td>
<td>200</td>
<td>μA</td>
<td>VDD = 3.0V, VPIN = VSS</td>
</tr>
<tr>
<td></td>
<td>VOL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D360</td>
<td>—</td>
<td></td>
<td>—</td>
<td>—</td>
<td>0.6</td>
<td>V</td>
<td>IOL = 10.0mA, VDD = 3.0V</td>
</tr>
<tr>
<td></td>
<td>VOH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D370</td>
<td>—</td>
<td></td>
<td>—</td>
<td>—</td>
<td>V</td>
<td>IOH = 6.0 mA, VDD = 3.0V</td>
<td></td>
</tr>
<tr>
<td>D380</td>
<td>—</td>
<td>All I/O pins</td>
<td>—</td>
<td>5</td>
<td>50</td>
<td>pF</td>
<td></td>
</tr>
</tbody>
</table>

† 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.

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 44-7: MEMORY PROGRAMMING SPECIFICATIONS

<table>
<thead>
<tr>
<th>Param No.</th>
<th>Sym.</th>
<th>Characteristic</th>
<th>Min.</th>
<th>Typ†</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEM20</td>
<td>ED</td>
<td>DataEE Byte Endurance</td>
<td>100k</td>
<td>—</td>
<td>—</td>
<td>E/W</td>
<td>-40°C ≤ TA ≤ +85°C</td>
</tr>
<tr>
<td>MEM21</td>
<td>TD_RET</td>
<td>Characteristic Retention</td>
<td>—</td>
<td>40</td>
<td>—</td>
<td>Year</td>
<td>Provided no other specifications are violated</td>
</tr>
<tr>
<td>MEM22</td>
<td>ND_REF</td>
<td>Total Erase/Write Cycles before Refresh</td>
<td>1M 500k</td>
<td>10M</td>
<td>—</td>
<td>E/W</td>
<td>-40°C ≤ TA ≤ +85°C</td>
</tr>
<tr>
<td>MEM23</td>
<td>VD_RW</td>
<td>VDD for Read or Erase/Write operation</td>
<td>VDDMIN</td>
<td>—</td>
<td>VDDMAX</td>
<td>V</td>
<td>—</td>
</tr>
<tr>
<td>MEM24</td>
<td>TD_BEW</td>
<td>Byte Erase and Write Cycle Time</td>
<td>—</td>
<td>4.0</td>
<td>5.0</td>
<td>ms</td>
<td>—</td>
</tr>
<tr>
<td>MEM30</td>
<td>EP</td>
<td>Memory Cell Endurance</td>
<td>10k</td>
<td>—</td>
<td>—</td>
<td>E/W</td>
<td>-40°C ≤ TA ≤ +85°C (Note 1)</td>
</tr>
<tr>
<td>MEM32</td>
<td>TP_RET</td>
<td>Characteristic Retention</td>
<td>—</td>
<td>40</td>
<td>—</td>
<td>Year</td>
<td>Provided no other specifications are violated</td>
</tr>
<tr>
<td>MEM33</td>
<td>VP_RD</td>
<td>VDD for Read operation</td>
<td>VDDMIN</td>
<td>—</td>
<td>VDDMAX</td>
<td>V</td>
<td>—</td>
</tr>
<tr>
<td>MEM34</td>
<td>VP_REW</td>
<td>VDD for Row Erase or Write operation</td>
<td>VDDMIN</td>
<td>—</td>
<td>VDDMAX</td>
<td>V</td>
<td>—</td>
</tr>
<tr>
<td>MEM35</td>
<td>TP_REW</td>
<td>Self-Timed Row Erase or Self-Timed Write</td>
<td>—</td>
<td>2.0</td>
<td>2.5</td>
<td>ms</td>
<td>—</td>
</tr>
</tbody>
</table>

† 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 44-8: THERMAL CHARACTERISTICS

**Standard Operating Conditions** (unless otherwise stated)

<table>
<thead>
<tr>
<th>Param No.</th>
<th>Sym.</th>
<th>Characteristic</th>
<th>Typ.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>TH01</td>
<td>θJA</td>
<td>Thermal Resistance Junction to Ambient</td>
<td>60</td>
<td>°C/W</td>
<td>28-pin SPDIP package</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>80</td>
<td>°C/W</td>
<td>28-pin SOIC package</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>90</td>
<td>°C/W</td>
<td>28-pin SSOP package</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>27.5</td>
<td>°C/W</td>
<td>28-pin UQFN 4x4 mm package</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>27.5</td>
<td>°C/W</td>
<td>28-pin QFN 6x6mm package</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>47.2</td>
<td>°C/W</td>
<td>40-pin PDIP package</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>46</td>
<td>°C/W</td>
<td>44-pin TQFP package</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>27.5</td>
<td>°C/W</td>
<td>28-pin UQFN 4x4 mm package</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>46</td>
<td>°C/W</td>
<td>44-pin TQFP package</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>24.4</td>
<td>°C/W</td>
<td>44-pin QFN 8x8mm package</td>
</tr>
<tr>
<td>TH02</td>
<td>θJC</td>
<td>Thermal Resistance Junction to Case</td>
<td>31.4</td>
<td>°C/W</td>
<td>28-pin SPDIP package</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>24</td>
<td>°C/W</td>
<td>28-pin SOIC package</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>24</td>
<td>°C/W</td>
<td>28-pin SSOP package</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>24</td>
<td>°C/W</td>
<td>28-pin UQFN 4x4 mm package</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>24</td>
<td>°C/W</td>
<td>28-pin QFN 6x6mm package</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>24.7</td>
<td>°C/W</td>
<td>40-pin PDIP package</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>14.5</td>
<td>°C/W</td>
<td>44-pin TQFP package</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>20</td>
<td>°C/W</td>
<td>44-pin QFN 8x8mm package</td>
</tr>
<tr>
<td>TH03</td>
<td>TJ MAX</td>
<td>Maximum Junction Temperature</td>
<td>150</td>
<td>°C</td>
<td></td>
</tr>
<tr>
<td>TH04</td>
<td>PD</td>
<td>Power Dissipation</td>
<td>—</td>
<td>W</td>
<td>PD = PINTERNAL + PI/O(3)</td>
</tr>
<tr>
<td>TH05</td>
<td>PINTERNAL</td>
<td>Internal Power Dissipation</td>
<td>—</td>
<td>W</td>
<td>PINTERNAL = IDD x VDD(1)</td>
</tr>
<tr>
<td>TH06</td>
<td>PI/O</td>
<td>I/O Power Dissipation</td>
<td>—</td>
<td>W</td>
<td>PI/O = Σ (IOL * VOL) + Σ (IOH * (VDD - VOH))</td>
</tr>
<tr>
<td>TH07</td>
<td>PDER</td>
<td>Derated Power</td>
<td>—</td>
<td>W</td>
<td>PDER = PDMAX (TJ - TA)/θJA(2)</td>
</tr>
</tbody>
</table>

**Note 1:** IDD is current to run the chip alone without driving any load on the output pins.

**Note 2:** TA = Ambient Temperature, TJ = Junction Temperature

**Note 3:** See absolute maximum ratings for total power dissipation.
44.4 AC Characteristics

FIGURE 44-4: LOAD CONDITIONS

Legend: CL=50 pF for all pins
FIGURE 44-5: CLOCK TIMING

TABLE 44-9: EXTERNAL CLOCK/OSCILLATOR TIMING REQUIREMENTS

<table>
<thead>
<tr>
<th>Param No.</th>
<th>Sym.</th>
<th>Characteristic</th>
<th>Min.</th>
<th>Typ†</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECL Oscillator</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OS1</td>
<td>F_{ECL}</td>
<td>Clock Frequency</td>
<td>—</td>
<td>—</td>
<td>500</td>
<td>kHz</td>
<td></td>
</tr>
<tr>
<td>OS2</td>
<td>T_{ECL, DC}</td>
<td>Clock Duty Cycle</td>
<td>40</td>
<td>—</td>
<td>60</td>
<td>%</td>
<td></td>
</tr>
<tr>
<td>ECM Oscillator</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OS3</td>
<td>F_{ECM}</td>
<td>Clock Frequency</td>
<td>—</td>
<td>—</td>
<td>8</td>
<td>MHz</td>
<td></td>
</tr>
<tr>
<td>OS4</td>
<td>T_{ECM, DC}</td>
<td>Clock Duty Cycle</td>
<td>40</td>
<td>—</td>
<td>60</td>
<td>%</td>
<td></td>
</tr>
<tr>
<td>ECH Oscillator</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OS5</td>
<td>F_{ECH}</td>
<td>Clock Frequency</td>
<td>—</td>
<td>—</td>
<td>64</td>
<td>MHz</td>
<td></td>
</tr>
<tr>
<td>OS6</td>
<td>T_{ECH, DC}</td>
<td>Clock Duty Cycle</td>
<td>40</td>
<td>—</td>
<td>60</td>
<td>%</td>
<td></td>
</tr>
<tr>
<td>LP Oscillator</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OS7</td>
<td>F_{LP}</td>
<td>Clock Frequency</td>
<td>—</td>
<td>—</td>
<td>100</td>
<td>kHz</td>
<td>Note 4</td>
</tr>
<tr>
<td>XT Oscillator</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OS8</td>
<td>F_{XT}</td>
<td>Clock Frequency</td>
<td>—</td>
<td>—</td>
<td>4</td>
<td>MHz</td>
<td>Note 4</td>
</tr>
<tr>
<td>HS Oscillator</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OS9</td>
<td>F_{HS}</td>
<td>Clock Frequency</td>
<td>—</td>
<td>—</td>
<td>20</td>
<td>MHz</td>
<td>Note 4</td>
</tr>
<tr>
<td>Secondary Oscillator</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OS10</td>
<td>F_{SEC}</td>
<td>Clock Frequency</td>
<td>32.4</td>
<td>32.768</td>
<td>33.1</td>
<td>kHz</td>
<td></td>
</tr>
<tr>
<td>System Oscillator</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OS20</td>
<td>F_{OSC}</td>
<td>System Clock Frequency</td>
<td>—</td>
<td>—</td>
<td>64</td>
<td>MHz</td>
<td>(Note 2, Note 3)</td>
</tr>
</tbody>
</table>

* 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 (TcY) 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 (Fosc) is selected by the “main clock switch controls” as described in Section 10.0 “Power-Saving Operation Modes”.

3: The system clock frequency (Fosc) must meet the voltage requirements defined in the Section 44.2 “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 44-9: EXTERNAL CLOCK/OSCILLATOR TIMING REQUIREMENTS (CONTINUED)

<table>
<thead>
<tr>
<th>Param No.</th>
<th>Sym.</th>
<th>Characteristic</th>
<th>Min.</th>
<th>Typ†</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS21</td>
<td>FCY</td>
<td>Instruction Frequency</td>
<td>—</td>
<td>FOSC/4</td>
<td>—</td>
<td>MHz</td>
<td></td>
</tr>
<tr>
<td>OS22</td>
<td>TCY</td>
<td>Instruction Period</td>
<td>62.5</td>
<td>1/F CY</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

* 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 (TCY) 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 (Fosc) is selected by the “main clock switch controls” as described in Section 10.0 “Power-Saving Operation Modes”.

3: The system clock frequency (Fosc) must meet the voltage requirements defined in the Section 44.2 “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 44-10: INTERNAL OSCILLATOR PARAMETERS(1)

<table>
<thead>
<tr>
<th>Param No.</th>
<th>Sym.</th>
<th>Characteristic</th>
<th>Min.</th>
<th>Typ†</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS50</td>
<td>HFOSC</td>
<td>Precision Calibrated HFINTOSC Frequency</td>
<td>—</td>
<td>4</td>
<td>8</td>
<td>MHz</td>
<td>(Note 2)</td>
</tr>
<tr>
<td>OS51</td>
<td>HFOSCLP</td>
<td>Low-Power Optimized HFINTOSC Frequency</td>
<td>0.93</td>
<td>1</td>
<td>2</td>
<td>MHz</td>
<td></td>
</tr>
<tr>
<td>OS53*</td>
<td>LFOSC</td>
<td>Internal LFINTOSC Frequency</td>
<td>—</td>
<td>31</td>
<td>—</td>
<td>kHz</td>
<td></td>
</tr>
<tr>
<td>OS54*</td>
<td>THFOS CST</td>
<td>HFINTOSC Wake-up from Sleep Start-up Time</td>
<td>—</td>
<td>11</td>
<td>20</td>
<td>µs</td>
<td>VREGPM = 0</td>
</tr>
<tr>
<td>OS56</td>
<td>TLFOS CST</td>
<td>LFINTOSC Wake-up from Sleep Start-up Time</td>
<td>—</td>
<td>0.2</td>
<td>—</td>
<td>ms</td>
<td>VREGPM = 1</td>
</tr>
</tbody>
</table>

* 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.

**Note 2:** See Figure 44-6: Precision Calibrated HFINTOSC Frequency Accuracy Over Device VDD and Temperature.

### FIGURE 44-6: PRECISION CALIBRATED HFINTOSC FREQUENCY ACCURACY OVER DEVICE VDD AND TEMPERATURE

![Graph showing frequency accuracy over VDD and temperature](image-url)
### TABLE 44-11: PLL SPECIFICATIONS

**Standard Operating Conditions (unless otherwise stated) V\text{DD} \geq 2.5V**

<table>
<thead>
<tr>
<th>Param No.</th>
<th>Sym.</th>
<th>Characteristic</th>
<th>Min.</th>
<th>Typ†</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLL01</td>
<td>FPLLIN</td>
<td>PLL Input Frequency Range</td>
<td>4</td>
<td>—</td>
<td>16</td>
<td>MHz</td>
<td></td>
</tr>
<tr>
<td>PLL02</td>
<td>FPLLOUT</td>
<td>PLL Output Frequency Range</td>
<td>16</td>
<td>—</td>
<td>64</td>
<td>MHz</td>
<td>Note 1</td>
</tr>
<tr>
<td>PLL03</td>
<td>TPLLST</td>
<td>PLL Lock Time from Start-up</td>
<td>—</td>
<td>200</td>
<td>—</td>
<td>(\mu)s</td>
<td></td>
</tr>
<tr>
<td>PLL04</td>
<td>FPLLJIT</td>
<td>PLL Output Frequency Stability (Jitter)</td>
<td>-0.25</td>
<td>—</td>
<td>0.25</td>
<td>%</td>
<td></td>
</tr>
</tbody>
</table>

* 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 F\text{OSC} requirements listed in Parameter D002.
TABLE 44-12: I/O AND CLKOUT TIMING SPECIFICATIONS

<table>
<thead>
<tr>
<th>Param No.</th>
<th>Sym.</th>
<th>Characteristic</th>
<th>Min.</th>
<th>Typ†</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>IO1*</td>
<td>T_CLKOUTH</td>
<td>CLKOUT rising edge delay (rising edge Fosc (Q1 cycle) to falling edge CLKOUT)</td>
<td>—</td>
<td>—</td>
<td>70</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>IO2*</td>
<td>T_CLKOUTL</td>
<td>CLKOUT falling edge delay (rising edge Fosc (Q3 cycle) to rising edge CLKOUT)</td>
<td>—</td>
<td>—</td>
<td>72</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>IO3*</td>
<td>T_IO_VALID</td>
<td>Port output valid time (rising edge Fosc (Q1 cycle) to port valid)</td>
<td>—</td>
<td>—</td>
<td>50</td>
<td>70</td>
<td>ns</td>
</tr>
<tr>
<td>IO4*</td>
<td>T_IO_SETUP</td>
<td>Port input setup time (Setup time before rising edge Fosc – Q2 cycle)</td>
<td>20</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>IO5*</td>
<td>T_IO_HOLD</td>
<td>Port input hold time (Hold time after rising edge Fosc – Q2 cycle)</td>
<td>50</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>IO6*</td>
<td>TIOR_SLRGEN</td>
<td>Port I/O rise time, slew rate enabled</td>
<td>—</td>
<td>25</td>
<td>—</td>
<td>ns</td>
<td>VDD = 3.0V</td>
</tr>
<tr>
<td>IO7*</td>
<td>TIOR_SLRDIS</td>
<td>Port I/O rise time, slew rate disabled</td>
<td>—</td>
<td>5</td>
<td>—</td>
<td>ns</td>
<td>VDD = 3.0V</td>
</tr>
<tr>
<td>IO8*</td>
<td>TIOF_SLRGEN</td>
<td>Port I/O fall time, slew rate enabled</td>
<td>—</td>
<td>25</td>
<td>—</td>
<td>ns</td>
<td>VDD = 3.0V</td>
</tr>
<tr>
<td>IO9*</td>
<td>TIOF_SLRDIS</td>
<td>Port I/O fall time, slew rate disabled</td>
<td>—</td>
<td>5</td>
<td>—</td>
<td>ns</td>
<td>VDD = 3.0V</td>
</tr>
<tr>
<td>IO10*</td>
<td>T_INT</td>
<td>INT pin high or low time to trigger an interrupt</td>
<td>25</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>IO11*</td>
<td>TIOC</td>
<td>Interrupt-on-Change minimum high or low time to trigger an interrupt</td>
<td>25</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

*These parameters are characterized but not tested.
FIGURE 44-8: 
RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER TIMING

Note 1: Asserted low.

FIGURE 44-9: 
BROWN-OUT RESET TIMING AND CHARACTERISTICS

Note 1: Delay depends on PWRTS<1:0> Configuration bits.
### TABLE 44-13: RESET, WDT, OSCILLATOR START-UP TIMER, POWER-UP TIMER, BROWN-OUT RESET AND LOW-POWER BROWN-OUT RESET SPECIFICATIONS

<table>
<thead>
<tr>
<th>Param No.</th>
<th>Sym.</th>
<th>Characteristic</th>
<th>Min.</th>
<th>Typ†</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>RST01*</td>
<td>TMCLR</td>
<td>MCLR Pulse Width Low to ensure Reset</td>
<td>2</td>
<td></td>
<td>—</td>
<td>μs</td>
<td></td>
</tr>
<tr>
<td>RST02*</td>
<td>I/O</td>
<td>I/O high-impedance from Reset detection</td>
<td>—</td>
<td>2</td>
<td>—</td>
<td>μs</td>
<td></td>
</tr>
<tr>
<td>RST03</td>
<td>TWD</td>
<td>Watchdog Timer Time-out Period</td>
<td>—</td>
<td>16</td>
<td>—</td>
<td>ms</td>
<td>PWRTS = 00</td>
</tr>
<tr>
<td>RST04*</td>
<td>TPWRT</td>
<td>Power-up Timer Period</td>
<td>—</td>
<td>1</td>
<td>—</td>
<td>ms</td>
<td>PWRTS = 01</td>
</tr>
<tr>
<td>RST05</td>
<td>16</td>
<td>Oscillator Start-up Timer Period(1,2)</td>
<td>—</td>
<td>1024</td>
<td>—</td>
<td>Tosc</td>
<td></td>
</tr>
<tr>
<td>RST06</td>
<td>VBO</td>
<td>Brown-out Reset Voltage(4)</td>
<td>2.7</td>
<td>2.85</td>
<td>3.0</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>RST07</td>
<td>BORV</td>
<td>Brown-out Reset Hysteresis</td>
<td>—</td>
<td>40</td>
<td>—</td>
<td>mV</td>
<td></td>
</tr>
<tr>
<td>RST08</td>
<td>TSOR</td>
<td>Brown-out Reset Response Time</td>
<td>—</td>
<td>3</td>
<td>—</td>
<td>ms</td>
<td></td>
</tr>
<tr>
<td>RST09</td>
<td>VLB</td>
<td>Low-Power Brown-out Reset Voltage</td>
<td>1.8</td>
<td>1.9</td>
<td>2.5</td>
<td>V</td>
<td>PIC18LFXXX</td>
</tr>
</tbody>
</table>

* 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.
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 44-14: HIGH/LOW-VOLTAGE DETECT CHARACTERISTICS

<table>
<thead>
<tr>
<th>Param No.</th>
<th>Symbol</th>
<th>Characteristic</th>
<th>Min.</th>
<th>Typ†</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>HLVD01</td>
<td>VDET</td>
<td>Voltage Detection</td>
<td>—</td>
<td>1.85</td>
<td>—</td>
<td>V</td>
<td>HLVDSEL&lt;3:0&gt;=0000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.85</td>
<td>2.06</td>
<td>—</td>
<td>V</td>
<td>HLVDSEL&lt;3:0&gt;=0001</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2.28</td>
<td>2.47</td>
<td>—</td>
<td>V</td>
<td>HLVDSEL&lt;3:0&gt;=0100</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2.57</td>
<td>2.78</td>
<td>—</td>
<td>V</td>
<td>HLVDSEL&lt;3:0&gt;=0101</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2.88</td>
<td>3.09</td>
<td>—</td>
<td>V</td>
<td>HLVDSEL&lt;3:0&gt;=0110</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3.40</td>
<td>3.60</td>
<td>—</td>
<td>V</td>
<td>HLVDSEL&lt;3:0&gt;=1000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3.71</td>
<td>3.91</td>
<td>—</td>
<td>V</td>
<td>HLVDSEL&lt;3:0&gt;=1010</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4.12</td>
<td>4.32</td>
<td>—</td>
<td>V</td>
<td>HLVDSEL&lt;3:0&gt;=1100</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4.32</td>
<td>4.63</td>
<td>—</td>
<td>V</td>
<td>HLVDSEL&lt;3:0&gt;=1110</td>
</tr>
</tbody>
</table>
**TABLE 44-15: ANALOG-TO-DIGITAL CONVERTER (ADC) ACCURACY SPECIFICATIONS**

Operating Conditions (unless otherwise stated)

<table>
<thead>
<tr>
<th>Param No.</th>
<th>Sym.</th>
<th>Characteristic</th>
<th>Min.</th>
<th>Typ†</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD01</td>
<td>NR</td>
<td>Resolution</td>
<td>—</td>
<td>—</td>
<td>12</td>
<td>bit</td>
<td></td>
</tr>
<tr>
<td>AD02</td>
<td>EIL</td>
<td>Integral Error</td>
<td>—</td>
<td>±0.1</td>
<td>±2.0</td>
<td>LSb</td>
<td>VDD = 3.0V, ADCREF+ = 3.0V, ADCREF- = 0V</td>
</tr>
<tr>
<td>AD03</td>
<td>EDL</td>
<td>Differential Error</td>
<td>—</td>
<td>±0.1</td>
<td>±1.0</td>
<td>LSb</td>
<td>VDD = 3.0V, ADCREF+ = 3.0V, ADCREF- = 0V</td>
</tr>
<tr>
<td>AD04</td>
<td>EOFF</td>
<td>Offset Error</td>
<td>—</td>
<td>0.5</td>
<td>6.0</td>
<td>LSb</td>
<td>VDD = 3.0V, ADCREF+ = 3.0V, ADCREF- = 0V</td>
</tr>
<tr>
<td>AD05</td>
<td>EG</td>
<td>Gain Error</td>
<td>—</td>
<td>±0.2</td>
<td>±6.0</td>
<td>LSb</td>
<td>VDD = 3.0V, ADCREF+ = 3.0V, ADCREF- = 0V</td>
</tr>
<tr>
<td>AD06</td>
<td>VADREF</td>
<td>ADC Reference Voltage (ADREF+ - ADREF-)</td>
<td>1.8</td>
<td>—</td>
<td>VDD</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>AD07</td>
<td>VAIN</td>
<td>Full-Scale Range</td>
<td>ADREF-</td>
<td>—</td>
<td>ADREF+</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>AD08</td>
<td>ZAIN</td>
<td>Recommended Impedance of Analog Voltage Source</td>
<td>—</td>
<td>1</td>
<td>—</td>
<td>kΩ</td>
<td></td>
</tr>
<tr>
<td>AD09</td>
<td>RVREF</td>
<td>ADC Voltage Reference Ladder Impedance</td>
<td>—</td>
<td>50</td>
<td>—</td>
<td>kΩ</td>
<td>Note 3</td>
</tr>
</tbody>
</table>

* 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.

**Note 2:** The ADC conversion result never decreases with an increase in the input and has no missing codes.

**Note 3:** This is the impedance seen by the VREF pads when the external reference pads are selected.
# TABLE 44-16: ANALOG-TO-DIGITAL CONVERTER (ADC) CONVERSION TIMING SPECIFICATIONS

<table>
<thead>
<tr>
<th>Param No.</th>
<th>Sym.</th>
<th>Characteristic</th>
<th>Min.</th>
<th>Typ†</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD20</td>
<td>TAD</td>
<td>ADC Clock Period</td>
<td>1</td>
<td>2</td>
<td>9</td>
<td>μs</td>
<td>Using FOSC as the ADC clock source ADCS = 1</td>
</tr>
<tr>
<td>AD21</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td>μs</td>
<td>Using FRC as the ADC clock source ADCS = 0</td>
</tr>
<tr>
<td>AD22</td>
<td>TCNV</td>
<td>Conversion Time(1)</td>
<td>14</td>
<td>16</td>
<td></td>
<td>—</td>
<td>Using FOSC as the ADC clock source ADCS = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>3</td>
<td></td>
<td>—</td>
<td>Using FRC as the ADC clock source ADCS = 0</td>
</tr>
<tr>
<td>AD24</td>
<td>THCD</td>
<td>Sample and Hold Capacitor Disconnect Time</td>
<td>2</td>
<td>3</td>
<td></td>
<td>—</td>
<td>Using FOSC as the ADC clock source ADCS = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>Using FRC as the ADC clock source ADCS = 0</td>
</tr>
</tbody>
</table>

* 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 44-10: ADC CONVERSION TIMING (ADC CLOCK FOSC-BASED)

![Diagram of ADC conversion timing]
FIGURE 44-11: ADC CONVERSION TIMING (ADC CLOCK FROM ADCRC)

- BSF ADCON0, GO
- ADC clk
- ADRES
- ADIF
- GO
- Sampling Stopped

Timings:
- 2 TCY
- 1 TCY
- Sampling Stopped

Legend:
- AD21
- AD22
- AD24

Notes:
- (1)
TABLE 44-17: COMPARATOR SPECIFICATIONS

**Operating Conditions** (unless otherwise stated)

\( V_{DD} = 3.0V, T_A = 25^\circ C \)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CM01</td>
<td>VIOFF</td>
<td>Input Offset Voltage</td>
<td>—</td>
<td>—</td>
<td>±40</td>
<td>mV</td>
<td>( V_{CM} = V_{DD}/2 )</td>
</tr>
<tr>
<td>CM02</td>
<td>VCM</td>
<td>Input Common Mode Range</td>
<td>GND</td>
<td>—</td>
<td>VDD</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>CM03</td>
<td>CMRR</td>
<td>Common Mode Input Rejection Ratio</td>
<td>—</td>
<td>50</td>
<td>—</td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td>CM04</td>
<td>VHYST</td>
<td>Comparator Hysteresis</td>
<td>10</td>
<td>25</td>
<td>40</td>
<td>mV</td>
<td></td>
</tr>
<tr>
<td>CM05</td>
<td>TRESP(^{(1)})</td>
<td>Response Time, Rising Edge</td>
<td>—</td>
<td>300</td>
<td>900</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Response Time, Falling Edge</td>
<td>—</td>
<td>220</td>
<td>500</td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

\(^{(1)}\) Response time measured with one comparator input at \( V_{DD}/2 \), while the other input transitions from \( V_{SS} \) to \( V_{DD} \).

**Note**

1: These parameters are characterized but not tested.
2: A mode change includes changing any of the control register values, including module enable.

TABLE 44-18: 5-BIT DAC SPECIFICATIONS

**Standard Operating Conditions** (unless otherwise stated)

\( V_{DD} = 3.0V, T_A = 25^\circ C \)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DSB01</td>
<td>VLSB</td>
<td>Step Size</td>
<td>—</td>
<td>—</td>
<td>(( V_{DACREF}+ - V_{DACREF}^- )/32)</td>
<td>—</td>
<td>V</td>
</tr>
<tr>
<td>DSB01</td>
<td>VACC</td>
<td>Absolute Accuracy</td>
<td>—</td>
<td>—</td>
<td>±0.5</td>
<td>( \text{LSb} )</td>
<td></td>
</tr>
<tr>
<td>DSB03*</td>
<td>RUNT</td>
<td>Unit Resistor Value</td>
<td>—</td>
<td>5000</td>
<td>—</td>
<td>( \Omega )</td>
<td></td>
</tr>
<tr>
<td>DSB04*</td>
<td>TST</td>
<td>Settling Time(^{(1)})</td>
<td>—</td>
<td>10</td>
<td>—</td>
<td>( \mu \text{s} )</td>
<td></td>
</tr>
</tbody>
</table>

\(^{(1)}\) 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 44-19: FIXED VOLTAGE REFERENCE (FVR) SPECIFICATIONS

**Standard Operating Conditions** (unless otherwise stated)

<table>
<thead>
<tr>
<th>Param No.</th>
<th>Symbol</th>
<th>Characteristic</th>
<th>Min.</th>
<th>Typ.</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>FVR01</td>
<td>FVR1</td>
<td>1x Gain (1.024V)</td>
<td>-4</td>
<td>—</td>
<td>+4</td>
<td>%</td>
<td>( V_{DD} \geq 2.5V, -40^\circ C ) to ( 85^\circ C )</td>
</tr>
<tr>
<td>FVR02</td>
<td>FVR2</td>
<td>2x Gain (2.048V)</td>
<td>-4</td>
<td>—</td>
<td>+4</td>
<td>%</td>
<td>( V_{DD} \geq 2.5V, -40^\circ C ) to ( 85^\circ C )</td>
</tr>
<tr>
<td>FVR03</td>
<td>FVR4</td>
<td>4x Gain (4.096V)</td>
<td>-5</td>
<td>—</td>
<td>+5</td>
<td>%</td>
<td>( V_{DD} \geq 4.75V, -40^\circ C ) to ( 85^\circ C )</td>
</tr>
<tr>
<td>FVR04</td>
<td>FVRST</td>
<td>FVR Start-up Time</td>
<td>—</td>
<td>25</td>
<td>—</td>
<td>us</td>
<td></td>
</tr>
</tbody>
</table>

TABLE 44-20: ZERO-CROSS DETECT (ZCD) SPECIFICATIONS

**Standard Operating Conditions** (unless otherwise stated)

\( V_{DD} = 3.0V, T_A = 25^\circ C \)

<table>
<thead>
<tr>
<th>Param No.</th>
<th>Sym.</th>
<th>Characteristics</th>
<th>Min.</th>
<th>Typ(^{†})</th>
<th>Max</th>
<th>Units</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZC01</td>
<td>VPINZC</td>
<td>Voltage on Zero Cross Pin</td>
<td>—</td>
<td>0.75</td>
<td>—</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>ZC02</td>
<td>ZCD_MAX</td>
<td>Maximum source or sink current</td>
<td>—</td>
<td>—</td>
<td>600</td>
<td>( \mu \text{A} )</td>
<td></td>
</tr>
<tr>
<td>ZC03</td>
<td>TRESPH</td>
<td>Response Time, Rising Edge</td>
<td>—</td>
<td>1</td>
<td>—</td>
<td>( \mu \text{s} )</td>
<td></td>
</tr>
<tr>
<td>ZC03</td>
<td>TRESPL</td>
<td>Response Time, Falling Edge</td>
<td>—</td>
<td>1</td>
<td>—</td>
<td>( \mu \text{s} )</td>
<td></td>
</tr>
</tbody>
</table>

\(^{†}\) 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 44-12: TIMER0 AND TIMER1 EXTERNAL CLOCK TIMINGS

TABLE 44-21: TIMER0 AND TIMER1 EXTERNAL CLOCK REQUIREMENTS

<table>
<thead>
<tr>
<th>Param No.</th>
<th>Sym.</th>
<th>Characteristic</th>
<th>Min.</th>
<th>Typ†</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>40*</td>
<td>T0H</td>
<td>T0CKI High Pulse Width</td>
<td>No Prescaler</td>
<td>0.5 Tcy + 20</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>With Prescaler</td>
<td>10</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>41*</td>
<td>T0L</td>
<td>T0CKI Low Pulse Width</td>
<td>No Prescaler</td>
<td>0.5 Tcy + 20</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>With Prescaler</td>
<td>10</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>42*</td>
<td>T0P</td>
<td>T0CKI Period</td>
<td>Greater of: 20 or Tcy + 40 N</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td>N = prescale value</td>
</tr>
<tr>
<td>45*</td>
<td>T1H</td>
<td>T1CKI High Time</td>
<td>Synchronous, No Prescaler</td>
<td>0.5 Tcy + 20</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Synchronous, with Prescaler</td>
<td>15</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Asynchronous</td>
<td>30</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>46*</td>
<td>T1L</td>
<td>T1CKI Low Time</td>
<td>Synchronous, No Prescaler</td>
<td>0.5 Tcy + 20</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Synchronous, with Prescaler</td>
<td>15</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Asynchronous</td>
<td>30</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>47*</td>
<td>T1P</td>
<td>T1CKI Input Period</td>
<td>Synchronous</td>
<td>Greater of: 30 or Tcy + 40 N</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Asynchronous</td>
<td>60</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>49*</td>
<td>TCKE2 TMR1</td>
<td>Delay from External Clock Edge to Timer Increment</td>
<td>2 Tosc</td>
<td>—</td>
<td>7 Tosc</td>
<td>—</td>
<td>Timers in Sync mode</td>
</tr>
</tbody>
</table>

* 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 44-13: CAPTURE/COMPARE/PWM TIMINGS (CCP)

![Diagram of CCPx timings](image)

**Note:** Refer to Figure 44-4 for load conditions.

### TABLE 44-22: CAPTURE/COMPARE/PWM REQUIREMENTS (CCP)

<table>
<thead>
<tr>
<th>Param No.</th>
<th>Sym.</th>
<th>Characteristic</th>
<th>Min.</th>
<th>Typ†</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC01*</td>
<td>TccL</td>
<td>CCPx Input Low Time</td>
<td>0.5TCY + 20</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td>No Prescaler</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>With Prescaler</td>
<td>20</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>CC02*</td>
<td>TccH</td>
<td>CCPx Input High Time</td>
<td>0.5TCY + 20</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td>No Prescaler</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>With Prescaler</td>
<td>20</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>CC03*</td>
<td>TccP</td>
<td>CCPx Input Period</td>
<td>3TCY + 40N</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td>N = prescale value</td>
</tr>
</tbody>
</table>

* 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.

---

**Standard Operating Conditions** (unless otherwise stated)
Operating Temperature: -40°C ≤ TA ≤ +125°C
FIGURE 44-14: SPI MASTER MODE TIMING (CKE = 0, SMP = 0)

Note: Refer to Figure 44-4 for load conditions.

FIGURE 44-15: SPI MASTER MODE TIMING (CKE = 1, SMP = 1)

Note: Refer to Figure 44-4 for load conditions.
FIGURE 44-16: SPI SLAVE MODE TIMING (CKE = 0)

FIGURE 44-17: SPI SLAVE MODE TIMING (CKE = 1)

Note: Refer to Figure 44-4 for load conditions.
## TABLE 44-23: SPI MODE REQUIREMENTS (MASTER MODE)

### Standard Operating Conditions (unless otherwise stated)

<table>
<thead>
<tr>
<th>Param No.</th>
<th>Symbol</th>
<th>Characteristic</th>
<th>Min.</th>
<th>Typ†</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>SCK Cycle Time (2x Prescaled)</td>
<td></td>
<td></td>
<td></td>
<td>ns</td>
<td>Transmit only mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>61</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>16(1)</td>
<td>—</td>
<td>MHz</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>95</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>10(1)</td>
<td>—</td>
<td>MHz</td>
<td></td>
</tr>
<tr>
<td>SP70*</td>
<td>TssL2sch,</td>
<td>SDO to SCK↓ or SCK↑ input</td>
<td></td>
<td></td>
<td></td>
<td>ns</td>
<td>FST = 0</td>
</tr>
<tr>
<td></td>
<td>TssL2scl</td>
<td></td>
<td>Tsk</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td>FST = 1</td>
</tr>
<tr>
<td>SP71*</td>
<td>Tsch</td>
<td>SCK output high time</td>
<td>0.5</td>
<td>Tsk</td>
<td>12</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.5</td>
<td>Tsk</td>
<td>12</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>SP72*</td>
<td>TscL</td>
<td>SCK output low time</td>
<td>0.5</td>
<td>Tsk</td>
<td>12</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.5</td>
<td>Tsk</td>
<td>12</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>SP73*</td>
<td>TdiV2sch,</td>
<td>Setup time of SDI data input to SCK edge</td>
<td>85</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TdiV2scl</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SP74*</td>
<td>TschH2diL,</td>
<td>Hold time of SDI data input to SCK edge</td>
<td>0</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td>CKE = 0, SMP = 1</td>
</tr>
<tr>
<td></td>
<td>TschL2diL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Hold time of SDI data input to final SCK</td>
<td>0.5</td>
<td>Tsk</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SP75*</td>
<td>TdoR</td>
<td>PDO data output rise time</td>
<td>—</td>
<td>10</td>
<td>25</td>
<td>ns</td>
<td>CKE = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SP76*</td>
<td>TdoF</td>
<td>PDO data output fall time</td>
<td>—</td>
<td>10</td>
<td>25</td>
<td>ns</td>
<td>CKE = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SP78*</td>
<td>TscR</td>
<td>SCK output rise time</td>
<td>—</td>
<td>10</td>
<td>25</td>
<td>ns</td>
<td>CKE = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SP79*</td>
<td>TscF</td>
<td>SCK output fall time</td>
<td>—</td>
<td>10</td>
<td>25</td>
<td>ns</td>
<td>CKE = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SP80*</td>
<td>TschH2doV,</td>
<td>SDO data output valid after SCK edge</td>
<td>—</td>
<td>15</td>
<td>15</td>
<td>ns</td>
<td>CKE = 1, CL = 20 pF</td>
</tr>
<tr>
<td></td>
<td>TschL2doV</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SP81*</td>
<td>TdoV2sch,</td>
<td>SDO data output valid to first SCK edge</td>
<td></td>
<td></td>
<td></td>
<td>ns</td>
<td>CKE = 1, CL = 20 pF</td>
</tr>
<tr>
<td></td>
<td>TdoV2scl</td>
<td></td>
<td>Tsk</td>
<td>10</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>SP82*</td>
<td>TssL2doV</td>
<td>SDO data output valid after SS↓ edge</td>
<td>—</td>
<td>50</td>
<td>—</td>
<td>ns</td>
<td>CKE = 1, CL = 20 pF</td>
</tr>
<tr>
<td>SP83*</td>
<td>TschH2ssh,</td>
<td>SS↑ after last SCK edge</td>
<td>0.5</td>
<td>Tsk</td>
<td>10</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td>TschL2ssh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SP84*</td>
<td>TssH2ssl</td>
<td>SS↑ to SS↓ edge</td>
<td>0.5</td>
<td>Tsk</td>
<td>10</td>
<td>—</td>
<td>ns</td>
</tr>
</tbody>
</table>

* 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 44-24: SPI Mode Requirements (Slave Mode)

#### Standard Operating Conditions (unless otherwise stated)

<table>
<thead>
<tr>
<th>Param No.</th>
<th>Symbol</th>
<th>Characteristic</th>
<th>Min.</th>
<th>Typ†</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TCK</td>
<td>SCK Total Cycle Time</td>
<td>47</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td>Receive only mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>20(1)</td>
<td>—</td>
<td>MHz</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>95</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td>Full duplex mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>10(1)</td>
<td>—</td>
<td>MHz</td>
<td></td>
</tr>
<tr>
<td>SP70*</td>
<td>TSSL2sCH, TSSL2sCL</td>
<td>SSL↓ to SCK↓ or SCK↑ input</td>
<td>0</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td>CKE = 0</td>
</tr>
<tr>
<td>SP71*</td>
<td>TSSC</td>
<td>SCK input high time</td>
<td>20</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td>CKE = 1</td>
</tr>
<tr>
<td>SP72*</td>
<td>TSL</td>
<td>SCK input low time</td>
<td>20</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>SP73*</td>
<td>TDIW2sCH, TDIW2sCL</td>
<td>Setup time of SDI data input to SCK edge</td>
<td>10</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>SP74*</td>
<td>TSH2oIL, TSL2oIL</td>
<td>Hold time of SDI data input to SCK edge</td>
<td>0</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>SP75*</td>
<td>TDO</td>
<td>SDO data output rise time</td>
<td>—</td>
<td>10</td>
<td>25</td>
<td>ns</td>
<td>CL = 50 pF</td>
</tr>
<tr>
<td>SP76*</td>
<td>TDOF</td>
<td>SDO data output fall time</td>
<td>—</td>
<td>10</td>
<td>25</td>
<td>ns</td>
<td>CL = 50 pF</td>
</tr>
<tr>
<td>SP77*</td>
<td>TSSH2oOZ</td>
<td>SSL↑ to SDO output high-impedance</td>
<td>—</td>
<td>—</td>
<td>85</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>SP80*</td>
<td>TSH2oOV, TSL2oOV</td>
<td>SDO data output valid after SCK edge</td>
<td>—</td>
<td>—</td>
<td>85</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>SP82*</td>
<td>TSSL2oOV</td>
<td>SDO data output valid after SSL↓ edge</td>
<td>—</td>
<td>—</td>
<td>85</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>SP83*</td>
<td>TSH2oSH, TSL2oSH</td>
<td>SSL↑ after SCK edge</td>
<td>20</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>SP84*</td>
<td>TSSH2oSL</td>
<td>SSL↑ to SSL↓ edge</td>
<td>47</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

* 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 44-18: \( \text{i}^{2}\text{C} \) BUS START/STOP BITS TIMING

Note: Refer to Figure 44-4 for load conditions.

TABLE 44-25: \( \text{i}^{2}\text{C} \) BUS START/STOP BITS REQUIREMENTS

<table>
<thead>
<tr>
<th>Param No.</th>
<th>Symbol</th>
<th>Characteristic</th>
<th>Min.</th>
<th>Typ</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP90*</td>
<td>TSU:STA</td>
<td>Start condition Setup time</td>
<td>100 kHz mode</td>
<td>4700</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>400 kHz mode</td>
<td>600</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 MHz mode</td>
<td>260</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>SP91*</td>
<td>THD:STA</td>
<td>Start condition Hold time</td>
<td>100 kHz mode</td>
<td>4000</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>400 kHz mode</td>
<td>600</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 MHz</td>
<td>260</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>SP92*</td>
<td>TSU:STO</td>
<td>Stop condition Setup time</td>
<td>100 kHz mode</td>
<td>4000</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>400 kHz mode</td>
<td>600</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 MHz</td>
<td>260</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>SP93</td>
<td>THD:STO</td>
<td>Stop condition Hold time</td>
<td>100 kHz mode</td>
<td>4700</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>400 kHz mode</td>
<td>1200</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 MHz</td>
<td>500</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
</tbody>
</table>

* These parameters are characterized but not tested.

Note: Refer to Figure 44-4 for load conditions.
<table>
<thead>
<tr>
<th>Param. No.</th>
<th>Symbol</th>
<th>Characteristic</th>
<th>Min.</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP100*</td>
<td>THIGH</td>
<td>Clock high time</td>
<td>100 kHz mode</td>
<td>4000</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>400 kHz mode</td>
<td>600</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 MHz module</td>
<td>260</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>SP101*</td>
<td>TLOW</td>
<td>Clock low time</td>
<td>100 kHz mode</td>
<td>4700</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>400 kHz mode</td>
<td>1300</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 MHz module</td>
<td>500</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>SP102*</td>
<td>TR</td>
<td>SDA and SCL rise time</td>
<td>100 kHz mode</td>
<td>—</td>
<td>1000</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>400 kHz mode</td>
<td>20</td>
<td>300</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 MHz module</td>
<td>—</td>
<td>120</td>
<td>ns</td>
</tr>
<tr>
<td>SP103*</td>
<td>TF</td>
<td>SDA and SCL fall time</td>
<td>100 kHz mode</td>
<td>—</td>
<td>250</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>400 kHz mode</td>
<td>20 X (VDD/5.5V)</td>
<td>250</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 MHz module</td>
<td>20 X (VDD/5.5V)</td>
<td>120</td>
<td>ns</td>
</tr>
<tr>
<td>SP106*</td>
<td>THD:DAT</td>
<td>Data input hold time</td>
<td>100 kHz mode</td>
<td>0</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>400 kHz mode</td>
<td>0</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 MHz module</td>
<td>0</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>SP107*</td>
<td>TSU:DAT</td>
<td>Data input setup time</td>
<td>100 kHz mode</td>
<td>250</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>400 kHz mode</td>
<td>100</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 MHz module</td>
<td>50</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>SP109*</td>
<td>TAA</td>
<td>Output valid from clock</td>
<td>100 kHz mode</td>
<td>—</td>
<td>3450</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>400 kHz mode</td>
<td>—</td>
<td>900</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 MHz module</td>
<td>—</td>
<td>450</td>
<td>ns</td>
</tr>
<tr>
<td>SP110*</td>
<td>TBUF</td>
<td>Bus free time</td>
<td>100 kHz mode</td>
<td>4700</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>400 kHz mode</td>
<td>1300</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 MHz module</td>
<td>500</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>SP111 C8</td>
<td>CB</td>
<td>Bus capacitive loading</td>
<td>—</td>
<td>400</td>
<td>pF</td>
<td></td>
</tr>
</tbody>
</table>

* 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.

**Note 2:** A Fast mode (400 kHz) I²C bus device can be used in a Standard mode (100 kHz) I²C bus system, but the requirement $T_{SU:DAT} \geq 250$ 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 $T_{R_{max}} + T_{SU:DAT} = 1000 + 250 = 1250$ ns (according to the Standard mode I²C bus specification), before the SCL line is released.
### TABLE 44-27: TEMPERATURE INDICATOR REQUIREMENTS

<table>
<thead>
<tr>
<th>Param No.</th>
<th>Symbol</th>
<th>Characteristic</th>
<th>Min.</th>
<th>Typ†</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>TS01*</td>
<td>TACQMIN</td>
<td>Minimum ADC Acquisition Time Delay</td>
<td>—</td>
<td>25</td>
<td>—</td>
<td>µs</td>
<td></td>
</tr>
<tr>
<td>TS02*</td>
<td>Mv</td>
<td>Voltage Sensitivity</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- High Range: —3.684 mV/°C, TSRNG = 1
- Low Range: —2.456 mV/°C, TSRNG = 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.
45.0 DC AND AC CHARACTERISTICS GRAPHS AND TABLES

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 VDD 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.

Note: 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 (mean + 3σ) or (mean - 3σ) respectively, where σ is a standard deviation, over each temperature range.
**Note:** Unless otherwise noted, \(V_{IN} = 5\text{V}\), \(F_{OSC} = 300\text{kHz}\), \(C_{IN} = 0.1\ \mu\text{F}\), \(T_A = 25\degree\text{C}\).

**FIGURE 45-1:** High Range Temperature Indicator Voltage Sensitivity Across Temperature

**FIGURE 45-2:** Low Range Temperature Indicator Voltage Sensitivity Across Temperature

**FIGURE 45-3:** Temperature Indicator Performance Over Temperature
46.0 PACKAGING INFORMATION

Package Marking Information

Legend:  
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  
† Pb-free JEDEC® designator for Matte Tin (Sn)  
* This package is Pb-free. The Pb-free JEDEC designator (†) 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

```
XXX...XX
XXX...XX
YYWWNNN
```

PIN 1

```
18F26K42
/ML e3
1526017
```

28-Lead UQFN (6x6x0.5 mm)

Example

```
X.XXXXXXXX
X.XXXXXXXX
YWWNNNN
```

PIN 1

```
PIC18
F26K42
/MX e3
526017
```

40-Lead PDIP (600 mil)

Example

```
X.XXXXXXXXXX
X.XXXXXXXXXX
X.XXXXXXXXXX
YWWNNNN
```

PIN 1

```
PIC18F45K42
/P e3
1526017
```

Legend:

- **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)

40-Lead UQFN (5x5x0.5 mm) Example

PIN 1

```
XX...X
XXXXXXX
XXXXXXX
YYYYYYX
YYWWNNN
```

44-Lead QFN (8x8x0.9 mm) Example

PIN 1

```
XX...X
XXXXXXXXXX
XXXXXXXXXX
XXXXXXXXXX
YYYYYYX
YYWWNNN
```

44-Lead TQFP (10x10x1 mm) Example

```
XX...X
XXXXXXXXXX
XXXXXXXXXX
XXXXXXXXXX
YYYYYYX
YYWWNNN
```

Legend:
- **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
- **@3**: Pb-free JEDEC® designator for Matte Tin (Sn)
- *****: This package is Pb-free. The Pb-free JEDEC designator (@3) 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)

48-Lead UQFN (6x6x0.5 mm)

Example

```
PIN 1
18F56K42
/MV 3e
1526017
```

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
* Pb-free JEDEC® designator for Matte Tin (Sn)
  This package is Pb-free. The Pb-free JEDEC designator (53) 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.
46.1 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

<table>
<thead>
<tr>
<th>Units</th>
<th>Dimension Limits</th>
<th>MIN</th>
<th>NOM</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Pins</td>
<td>N</td>
<td>28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pitch</td>
<td>e</td>
<td>.100 BSC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Top to Seating Plane</td>
<td>A</td>
<td>–</td>
<td>–</td>
<td>.200</td>
</tr>
<tr>
<td>Molded Package Thickness</td>
<td>A2</td>
<td>.120</td>
<td>.135</td>
<td>.150</td>
</tr>
<tr>
<td>Base to Seating Plane</td>
<td>A1</td>
<td>.015</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Shoulder to Shoulder Width</td>
<td>E</td>
<td>.290</td>
<td>.310</td>
<td>.335</td>
</tr>
<tr>
<td>Molded Package Width</td>
<td>E1</td>
<td>.240</td>
<td>.285</td>
<td>.295</td>
</tr>
<tr>
<td>Overall Length</td>
<td>D</td>
<td>1.345</td>
<td>1.365</td>
<td>1.400</td>
</tr>
<tr>
<td>Tip to Seating Plane</td>
<td>L</td>
<td>.110</td>
<td>.130</td>
<td>.150</td>
</tr>
<tr>
<td>Lead Thickness</td>
<td>c</td>
<td>.008</td>
<td>.010</td>
<td>.015</td>
</tr>
<tr>
<td>Upper Lead Width</td>
<td>b1</td>
<td>.040</td>
<td>.050</td>
<td>.070</td>
</tr>
<tr>
<td>Lower Lead Width</td>
<td>b</td>
<td>.014</td>
<td>.018</td>
<td>.022</td>
</tr>
<tr>
<td>Overall Row Spacing</td>
<td>eB</td>
<td>–</td>
<td>–</td>
<td>.430</td>
</tr>
</tbody>
</table>

Notes:
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic.
3. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" per side.
4. Dimensioning and tolerancing per ASME Y14.5M.
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
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
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

<table>
<thead>
<tr>
<th>Units</th>
<th>MILLIMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Pins</td>
<td>N</td>
</tr>
<tr>
<td>Pitch</td>
<td>e</td>
</tr>
<tr>
<td>Overall Height</td>
<td>A</td>
</tr>
<tr>
<td>Molded Package Thickness</td>
<td>A2</td>
</tr>
<tr>
<td>Standoff</td>
<td>§</td>
</tr>
<tr>
<td>Overall Width</td>
<td>E</td>
</tr>
<tr>
<td>Molded Package Width</td>
<td>E1</td>
</tr>
<tr>
<td>Overall Length</td>
<td>D</td>
</tr>
<tr>
<td>Chamfer (Optional)</td>
<td>h</td>
</tr>
<tr>
<td>Foot Length</td>
<td>L</td>
</tr>
<tr>
<td>Footprint</td>
<td>L1</td>
</tr>
<tr>
<td>Lead Angle</td>
<td>θ</td>
</tr>
<tr>
<td>Foot Angle</td>
<td>φ</td>
</tr>
<tr>
<td>Lead Thickness</td>
<td>c</td>
</tr>
<tr>
<td>Lead Width</td>
<td>b</td>
</tr>
<tr>
<td>Mold Draft Angle Top</td>
<td>α</td>
</tr>
<tr>
<td>Mold Draft Angle Bottom</td>
<td>β</td>
</tr>
</tbody>
</table>

Notes:
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic
3. 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.
4. 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.
5. Datums A & B to be determined at Datum H.
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

<table>
<thead>
<tr>
<th>Units</th>
<th>MILLIMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimension</td>
<td></td>
</tr>
<tr>
<td>Limits</td>
<td>MIN</td>
</tr>
<tr>
<td>Contact Pitch</td>
<td>E</td>
</tr>
<tr>
<td>Contact Pad Spacing</td>
<td>C</td>
</tr>
<tr>
<td>Contact Pad Width (X28)</td>
<td>X</td>
</tr>
<tr>
<td>Contact Pad Length (X28)</td>
<td>Y</td>
</tr>
<tr>
<td>Distance Between Pads</td>
<td>Gx</td>
</tr>
<tr>
<td>Distance Between Pads</td>
<td>G</td>
</tr>
</tbody>
</table>

Notes:
1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2052A
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

**Dimensions Table:***

<table>
<thead>
<tr>
<th>Units</th>
<th>MILLIMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimension Limits</td>
<td>MIN</td>
</tr>
<tr>
<td>Number of Pins</td>
<td>N</td>
</tr>
<tr>
<td>Pitch</td>
<td>e</td>
</tr>
<tr>
<td>Overall Height</td>
<td>A</td>
</tr>
<tr>
<td>Molded Package Thickness</td>
<td>A2</td>
</tr>
<tr>
<td>Standoff</td>
<td>A1</td>
</tr>
<tr>
<td>Overall Width</td>
<td>E</td>
</tr>
<tr>
<td>Molded Package Width</td>
<td>E1</td>
</tr>
<tr>
<td>Overall Length</td>
<td>D</td>
</tr>
<tr>
<td>Foot Length</td>
<td>L</td>
</tr>
<tr>
<td>Footprint</td>
<td>L1</td>
</tr>
<tr>
<td>Lead Thickness</td>
<td>c</td>
</tr>
<tr>
<td>Foot Angle</td>
<td>φ</td>
</tr>
<tr>
<td>Lead Width</td>
<td>b</td>
</tr>
</tbody>
</table>

**Notes:**
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.20 mm per side.
3. 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
Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging

<table>
<thead>
<tr>
<th></th>
<th>Units</th>
<th>MILLIMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MIN</td>
<td>NOM</td>
</tr>
<tr>
<td>Contact Pitch</td>
<td>E</td>
<td>0.65 BSC</td>
</tr>
<tr>
<td>Contact Pad Spacing</td>
<td>C</td>
<td>7.20</td>
</tr>
<tr>
<td>Contact Pad Width (X28)</td>
<td>X1</td>
<td>0.45</td>
</tr>
<tr>
<td>Contact Pad Length (X28)</td>
<td>Y1</td>
<td>1.75</td>
</tr>
<tr>
<td>Distance Between Pads</td>
<td>G</td>
<td>0.20</td>
</tr>
</tbody>
</table>

Notes:
1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
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
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

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Units</th>
<th>MILIMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Pins</td>
<td>N</td>
<td>28</td>
</tr>
<tr>
<td>Pitch</td>
<td>e</td>
<td>0.65 BSC</td>
</tr>
<tr>
<td>Overall Height</td>
<td>A</td>
<td>0.80 0.90 1.00</td>
</tr>
<tr>
<td>Standoff</td>
<td>A1</td>
<td>0.00 0.02 0.05</td>
</tr>
<tr>
<td>Terminal Thickness</td>
<td>A3</td>
<td>0.20 REF</td>
</tr>
<tr>
<td>Overall Width</td>
<td>E</td>
<td>6.00 BSC</td>
</tr>
<tr>
<td>Exposed Pad Width</td>
<td>E2</td>
<td>3.65 3.70 4.20</td>
</tr>
<tr>
<td>Overall Length</td>
<td>D</td>
<td>6.00 BSC</td>
</tr>
<tr>
<td>Exposed Pad Length</td>
<td>D2</td>
<td>3.65 3.70 4.20</td>
</tr>
<tr>
<td>Terminal Width</td>
<td>b</td>
<td>0.23 0.30 0.35</td>
</tr>
<tr>
<td>Terminal Length</td>
<td>L</td>
<td>0.50 0.55 0.70</td>
</tr>
<tr>
<td>Terminal-to-Exposed Pad</td>
<td>K</td>
<td>0.20 - -</td>
</tr>
</tbody>
</table>

**Notes:**
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated
3. 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.
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

<table>
<thead>
<tr>
<th>Units</th>
<th>MILLIMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimension</td>
<td>MIN</td>
</tr>
<tr>
<td>Contact Pitch</td>
<td>E</td>
</tr>
<tr>
<td>Optional Center Pad Width</td>
<td>W2</td>
</tr>
<tr>
<td>Optional Center Pad Length</td>
<td>T2</td>
</tr>
<tr>
<td>Contact Pad Spacing</td>
<td>C1</td>
</tr>
<tr>
<td>Contact Pad Spacing</td>
<td>C2</td>
</tr>
<tr>
<td>Contact Pad Width (X28)</td>
<td>X1</td>
</tr>
<tr>
<td>Contact Pad Length (X28)</td>
<td>Y1</td>
</tr>
<tr>
<td>Distance Between Pads</td>
<td>G</td>
</tr>
</tbody>
</table>

Notes:
1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
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](http://www.microchip.com/packaging)
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

<table>
<thead>
<tr>
<th>Units</th>
<th>MILLIMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimension Limits</td>
<td>MIN</td>
</tr>
<tr>
<td>Number of Pins</td>
<td>N</td>
</tr>
<tr>
<td>Pitch</td>
<td>e</td>
</tr>
<tr>
<td>Overall Height</td>
<td>A</td>
</tr>
<tr>
<td>Standoff</td>
<td>A1</td>
</tr>
<tr>
<td>Terminal Thickness</td>
<td>(A3)</td>
</tr>
<tr>
<td>Overall Width</td>
<td>E</td>
</tr>
<tr>
<td>Exposed Pad Width</td>
<td>E2</td>
</tr>
<tr>
<td>Overall Length</td>
<td>D</td>
</tr>
<tr>
<td>Exposed Pad Length</td>
<td>D2</td>
</tr>
<tr>
<td>Terminal Width</td>
<td>b</td>
</tr>
<tr>
<td>Corner Pad</td>
<td>b1</td>
</tr>
<tr>
<td>Corner Pad, Metal Free Zone</td>
<td>b2</td>
</tr>
<tr>
<td>Terminal Length</td>
<td>L</td>
</tr>
<tr>
<td>Terminal-to-Exposed Pad</td>
<td>K</td>
</tr>
</tbody>
</table>

**Notes:**
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated
3. 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.
4. Outermost portions of corner structures may vary slightly.
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

<table>
<thead>
<tr>
<th>Units</th>
<th>Dimension Limits</th>
<th>MILLIMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>MIN</td>
</tr>
<tr>
<td>Contact Pitch</td>
<td>E</td>
<td>0.65</td>
</tr>
<tr>
<td>Optional Center Pad Width</td>
<td>W1</td>
<td></td>
</tr>
<tr>
<td>Optional Center Pad Length</td>
<td>T2</td>
<td></td>
</tr>
<tr>
<td>Contact Pad Spacing</td>
<td>C1</td>
<td>5.70</td>
</tr>
<tr>
<td>Contact Pad Spacing</td>
<td>C2</td>
<td>5.70</td>
</tr>
<tr>
<td>Contact Pad Width (X28)</td>
<td>X1</td>
<td></td>
</tr>
<tr>
<td>Contact Pad Length (X28)</td>
<td>Y1</td>
<td></td>
</tr>
<tr>
<td>Corner Pad Width (X4)</td>
<td>X2</td>
<td></td>
</tr>
<tr>
<td>Corner Pad Length (X4)</td>
<td>Y2</td>
<td></td>
</tr>
<tr>
<td>Distance Between Pads</td>
<td>G</td>
<td>0.20</td>
</tr>
</tbody>
</table>

Notes:
1. Dimensioning and tolerancing per ASME Y14.5M  
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2209B
40-Lead Plastic Dual In-Line (P) – 600 mil Body [PDIP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging

<table>
<thead>
<tr>
<th>Units</th>
<th>INCHES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimension Limits</td>
<td>MIN</td>
</tr>
<tr>
<td>Number of Pins</td>
<td>N</td>
</tr>
<tr>
<td>Pitch</td>
<td>e</td>
</tr>
<tr>
<td>Top to Seating Plane</td>
<td>A</td>
</tr>
<tr>
<td>Molded Package Thickness</td>
<td>A2</td>
</tr>
<tr>
<td>Base to Seating Plane</td>
<td>A1</td>
</tr>
<tr>
<td>Shoulder to Shoulder Width</td>
<td>E</td>
</tr>
<tr>
<td>Molded Package Width</td>
<td>E1</td>
</tr>
<tr>
<td>Overall Length</td>
<td>D</td>
</tr>
<tr>
<td>Tip to Seating Plane</td>
<td>L</td>
</tr>
<tr>
<td>Lead Thickness</td>
<td>c</td>
</tr>
<tr>
<td>Upper Lead Width</td>
<td>b1</td>
</tr>
<tr>
<td>Lower Lead Width</td>
<td>b</td>
</tr>
<tr>
<td>Overall Row Spacing §</td>
<td>eB</td>
</tr>
</tbody>
</table>

**Notes:**
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic.
3. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010” per side.
4. Dimensioning and tolerancing per ASME Y14.5M.

BSC: Basic Dimension. Theoretically exact value shown without tolerances.
40-Lead Ultra Thin Plastic Quad Flat, No Lead Package (MV) – 5x5x0.5 mm Body [UQFN]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at [http://www.microchip.com/packaging](http://www.microchip.com/packaging)
40-Lead Ultra Thin Plastic Quad Flat, No Lead Package (MV) – 5x5x0.5 mm Body [UQFN]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging

![Diagram of the package](image)

<table>
<thead>
<tr>
<th>Units</th>
<th>MILLIMETERS</th>
<th>MIN</th>
<th>NOM</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Pins</td>
<td>N</td>
<td>40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pitch</td>
<td>e</td>
<td></td>
<td>0.40 BSC</td>
<td></td>
</tr>
<tr>
<td>Overall Height</td>
<td>A</td>
<td>0.45</td>
<td>0.50</td>
<td>0.55</td>
</tr>
<tr>
<td>Standoff</td>
<td>A1</td>
<td>0.00</td>
<td>0.02</td>
<td>0.05</td>
</tr>
<tr>
<td>Contact Thickness</td>
<td>A3</td>
<td></td>
<td>0.127 REF</td>
<td></td>
</tr>
<tr>
<td>Overall Width</td>
<td>E</td>
<td></td>
<td>5.00 BSC</td>
<td></td>
</tr>
<tr>
<td>Exposed Pad Width</td>
<td>E2</td>
<td>3.60</td>
<td>3.70</td>
<td>3.80</td>
</tr>
<tr>
<td>Overall Length</td>
<td>D</td>
<td></td>
<td>5.00 BSC</td>
<td></td>
</tr>
<tr>
<td>Exposed Pad Length</td>
<td>D2</td>
<td>3.60</td>
<td>3.70</td>
<td>3.80</td>
</tr>
<tr>
<td>Contact Width</td>
<td>b</td>
<td>0.15</td>
<td>0.20</td>
<td>0.25</td>
</tr>
<tr>
<td>Contact Length</td>
<td>L</td>
<td>0.30</td>
<td>0.40</td>
<td>0.50</td>
</tr>
<tr>
<td>Contact-to-Exposed Pad</td>
<td>K</td>
<td>0.20</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Notes:**
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated.
3. 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 C04-156A Sheet 2 of 2
40-Lead Plastic Ultra Thin Quad Flat, No Lead Package (MV) - 5x5 mm Body [UQFN]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging

<table>
<thead>
<tr>
<th>Units</th>
<th>MILIMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dimension Limits</strong></td>
<td>MIN</td>
</tr>
<tr>
<td>Contact Pitch</td>
<td>E</td>
</tr>
<tr>
<td>Optional Center Pad Width</td>
<td>W2</td>
</tr>
<tr>
<td>Optional Center Pad Length</td>
<td>T2</td>
</tr>
<tr>
<td>Contact Pad Spacing</td>
<td>C1</td>
</tr>
<tr>
<td>Contact Pad Spacing</td>
<td>C2</td>
</tr>
<tr>
<td>Contact Pad Width (X40)</td>
<td>X1</td>
</tr>
<tr>
<td>Contact Pad Length (X40)</td>
<td>Y1</td>
</tr>
<tr>
<td>Distance Between Pads</td>
<td>G</td>
</tr>
</tbody>
</table>

Notes:
1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2156B
44-Lead Plastic Quad Flat, No Lead Package (ML) - 8x8 mm Body [QFN or VQFN]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging
44-Lead Plastic Quad Flat, No Lead Package (ML) - 8x8 mm Body [QFN or VQFN]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging

<table>
<thead>
<tr>
<th>Units</th>
<th>MILLIMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimension Limit</td>
<td>MIN</td>
</tr>
<tr>
<td>Number of Pins</td>
<td>N</td>
</tr>
<tr>
<td>Pitch</td>
<td>e</td>
</tr>
<tr>
<td>Overall Height</td>
<td>A</td>
</tr>
<tr>
<td>Standoff</td>
<td>A1</td>
</tr>
<tr>
<td>Terminal Thickness</td>
<td>A3</td>
</tr>
<tr>
<td>Overall Width</td>
<td>E</td>
</tr>
<tr>
<td>Exposed Pad Width</td>
<td>E2</td>
</tr>
<tr>
<td>Overall Length</td>
<td>D</td>
</tr>
<tr>
<td>Exposed Pad Length</td>
<td>D2</td>
</tr>
<tr>
<td>Terminal Width</td>
<td>b</td>
</tr>
<tr>
<td>Terminal Length</td>
<td>L</td>
</tr>
<tr>
<td>Terminal-to-Exposed-Pad</td>
<td>K</td>
</tr>
</tbody>
</table>

Notes:
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated.
3. 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 C04-103D Sheet 2 of 2
44-Lead Plastic Quad Flat, No Lead Package (ML) - 8x8 mm Body [QFN or VQFN]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com-packaging

---

**RECOMMENDED LAND PATTERN**

<table>
<thead>
<tr>
<th>Units</th>
<th>MILLIMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dimension Limits</strong></td>
<td></td>
</tr>
<tr>
<td>Contact Pitch</td>
<td>E 0.65 BSC</td>
</tr>
<tr>
<td>Optional Center Pad Width</td>
<td>X2 6.60</td>
</tr>
<tr>
<td>Optional Center Pad Length</td>
<td>Y2 6.60</td>
</tr>
<tr>
<td>Contact Pad Spacing</td>
<td>C1 8.00</td>
</tr>
<tr>
<td>Contact Pad Spacing</td>
<td>C2 8.00</td>
</tr>
<tr>
<td>Contact Pad Width (X44)</td>
<td>X1 0.35</td>
</tr>
<tr>
<td>Contact Pad Length (X44)</td>
<td>Y1 0.85</td>
</tr>
<tr>
<td>Contact Pad to Contact Pad (X40)</td>
<td>G1 0.30</td>
</tr>
<tr>
<td>Contact Pad to Center Pad (X44)</td>
<td>G2 0.28</td>
</tr>
<tr>
<td>Thermal Via Diameter</td>
<td>ØV 0.33</td>
</tr>
<tr>
<td>Thermal Via Pitch</td>
<td>EV 1.20</td>
</tr>
</tbody>
</table>

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
2. For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

Microchip Technology Drawing No. C04-2103C
44-Lead Plastic Thin Quad Flatpack (PT) - 10x10x1.0 mm Body [TQFP]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging
44-Lead Plastic Thin Quad Flatpack (PT) - 10x10x1.0 mm Body [TQFP]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging

<table>
<thead>
<tr>
<th>Dimension Limits</th>
<th>MIN</th>
<th>NOM</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Leads</td>
<td>N</td>
<td>44</td>
<td></td>
</tr>
<tr>
<td>Lead Pitch</td>
<td>e</td>
<td>0.80 BSC</td>
<td></td>
</tr>
<tr>
<td>Overall Height</td>
<td>A</td>
<td>-</td>
<td>1.20</td>
</tr>
<tr>
<td>Standoff</td>
<td>A1</td>
<td>0.05</td>
<td>0.15</td>
</tr>
<tr>
<td>Molded Package Thickness</td>
<td>A2</td>
<td>0.95</td>
<td>1.00</td>
</tr>
<tr>
<td>Overall Width</td>
<td>E</td>
<td>12.00 BSC</td>
<td></td>
</tr>
<tr>
<td>Molded Package Width</td>
<td>E1</td>
<td>10.00 BSC</td>
<td></td>
</tr>
<tr>
<td>Overall Length</td>
<td>D</td>
<td>12.00 BSC</td>
<td></td>
</tr>
<tr>
<td>Molded Package Length</td>
<td>D1</td>
<td>10.00 BSC</td>
<td></td>
</tr>
<tr>
<td>Lead Width</td>
<td>b</td>
<td>0.30</td>
<td>0.37</td>
</tr>
<tr>
<td>Lead Thickness</td>
<td>c</td>
<td>0.09</td>
<td>-</td>
</tr>
<tr>
<td>Lead Length</td>
<td>L</td>
<td>0.45</td>
<td>0.60</td>
</tr>
<tr>
<td>Footprint</td>
<td>L1</td>
<td>1.00 REF</td>
<td></td>
</tr>
<tr>
<td>Foot Angle</td>
<td>θ</td>
<td>0°</td>
<td>3.5°</td>
</tr>
</tbody>
</table>

Notes:
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Exact shape of each corner is optional.
3. 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.
44-Lead Plastic Thin Quad Flatpack (PT) - 10X10X1 mm Body, 2.00 mm Footprint [TQFP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging

<table>
<thead>
<tr>
<th>Units</th>
<th>MILLIMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimension Limits</td>
<td>MIN</td>
</tr>
<tr>
<td>Contact Pitch</td>
<td>E</td>
</tr>
<tr>
<td>Contact Pad Spacing</td>
<td>C1</td>
</tr>
<tr>
<td>Contact Pad Spacing</td>
<td>C2</td>
</tr>
<tr>
<td>Contact Pad Width (X44)</td>
<td>X1</td>
</tr>
<tr>
<td>Contact Pad Length (X44)</td>
<td>Y1</td>
</tr>
<tr>
<td>Distance Between Pads</td>
<td>G</td>
</tr>
</tbody>
</table>

Notes:
1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2076B
48-Lead Plastic Ultra Thin Quad Flat, No Lead Package (MV) – 6x6x0.5 mm Body [UQFN]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging
### 48-Lead Plastic Ultra Thin Quad Flat, No Lead Package (MV) – 6x6x0.5 mm Body [UQFN]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at [http://www.microchip.com/packaging](http://www.microchip.com/packaging)

---

**Units** | **MILLIMETERS**
--- | ---
**Dimension Limits** | **MIN** | **NOM** | **MAX**
--- | --- | --- | ---
Number of Pins | N | 48 | 48 |
Pitch | e | 0.40 BSC | 0.40 BSC |
Overall Height | A | 0.45 | 0.50 | 0.55 |
Standoff | A1 | 0.00 | 0.02 | 0.05 |
Contact Thickness | A3 | 0.127 REF | 0.127 REF |
Overall Width | E | 6.00 BSC | 6.00 BSC |
Exposed Pad Width | E2 | 4.45 | 4.60 | 4.75 |
Exposed Pad Length | D2 | 4.45 | 4.60 | 4.75 |
Contact Width | b | 0.15 | 0.20 | 0.25 |
Contact Length | L | 0.30 | 0.40 | 0.50 |
Contact-to-Exposed Pad | K | 0.20 | - | - |

**Notes:**
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated.
3. 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  C04-153A Sheet 2 of 2
PIC18(L)F26/27/45/46/47/55/56/57K42

48-Lead Ultra Thin Plastic Quad Flat, No Lead Package (MV) - 6x6 mm Body [UQFN]
With 0.40 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**

---

<table>
<thead>
<tr>
<th>Units</th>
<th>MILLIMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimension Limits</td>
<td>MIN</td>
</tr>
<tr>
<td>Contact Pitch</td>
<td>E</td>
</tr>
<tr>
<td>Optional Center Pad Width</td>
<td>W2</td>
</tr>
<tr>
<td>Optional Center Pad Length</td>
<td>T2</td>
</tr>
<tr>
<td>Contact Pad Spacing</td>
<td>C1</td>
</tr>
<tr>
<td>Contact Pad Spacing</td>
<td>C2</td>
</tr>
<tr>
<td>Contact Pad Width (X28)</td>
<td>X1</td>
</tr>
<tr>
<td>Contact Pad Length (X28)</td>
<td>Y1</td>
</tr>
<tr>
<td>Distance Between Pads</td>
<td>G</td>
</tr>
</tbody>
</table>

**Notes:**
1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2153A
48-Lead Thin Quad Flatpack (PT) - 7x7x1.0 mm Body [TQFP] With Exposed Pad

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging
### 48-Lead Thin Quad Flatpack (PT) - 7x7x1.0 mm Body [TQFP] With Exposed Pad

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at [http://www.microchip.com/packaging](http://www.microchip.com/packaging)

#### 2.1.4 

**BSC:** Basic Dimension. Theoretically exact value shown without tolerances.

**REF:** Reference Dimension, usually without tolerance, for information purposes only.

<table>
<thead>
<tr>
<th>Units</th>
<th>MILLIMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Leads</td>
<td>N 48</td>
</tr>
<tr>
<td>Lead Pitch</td>
<td>e 0.50 BSC</td>
</tr>
<tr>
<td>Overall Height</td>
<td>A 0.95 -</td>
</tr>
<tr>
<td>Standoff</td>
<td>A1 0.05 -</td>
</tr>
<tr>
<td>Molded Package Thickness</td>
<td>A2 1.00 0.60 0.75</td>
</tr>
<tr>
<td>Foot Length</td>
<td>L 0.45 0.60 0.75</td>
</tr>
<tr>
<td>Footprint</td>
<td>L1 1.00 REF</td>
</tr>
<tr>
<td>Foot Angle</td>
<td>ϕ 3.5° 7°</td>
</tr>
<tr>
<td>Overall Width</td>
<td>E 9.00 BSC</td>
</tr>
<tr>
<td>Overall Length</td>
<td>D 9.00 BSC</td>
</tr>
<tr>
<td>Molded Package Width</td>
<td>E1 7.00 BSC</td>
</tr>
<tr>
<td>Molded Package Length</td>
<td>D1 7.00 BSC</td>
</tr>
<tr>
<td>Exposed Pad Width</td>
<td>E2 3.50 BSC</td>
</tr>
<tr>
<td>Exposed Pad Length</td>
<td>D2 3.50 BSC</td>
</tr>
<tr>
<td>Lead Thickness</td>
<td>c 0.09 - 0.16</td>
</tr>
<tr>
<td>Lead Width</td>
<td>b 0.17 0.22 0.27</td>
</tr>
<tr>
<td>Mold Draft Angle Top</td>
<td>α 11° 12° 13°</td>
</tr>
<tr>
<td>Mold Draft Angle Bottom</td>
<td>β 11° 12° 13°</td>
</tr>
</tbody>
</table>

**Notes:**

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Chamfers at corners are optional; size may vary.
3. Dimensions D1 and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.25mm per side.
4. 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.
48-Lead Thin Quad Flatpack (PT) - 7x7x1.0 mm Body [TQFP] With Thermal Tab

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging

---

**RECOMMENDED LAND PATTERN**

<table>
<thead>
<tr>
<th>Units</th>
<th>MILLIMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimension Limits</td>
<td>MIN</td>
</tr>
<tr>
<td>Contact Pitch</td>
<td>E</td>
</tr>
<tr>
<td>Optional Center Tab Width</td>
<td>X2</td>
</tr>
<tr>
<td>Optional Center Tab Length</td>
<td>Y2</td>
</tr>
<tr>
<td>Contact Pad Spacing</td>
<td>C1</td>
</tr>
<tr>
<td>Contact Pad Spacing</td>
<td>C2</td>
</tr>
<tr>
<td>Contact Pad Width (X48)</td>
<td>X1</td>
</tr>
<tr>
<td>Contact Pad Length (X48)</td>
<td>Y1</td>
</tr>
</tbody>
</table>

**Notes:**
1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2183A
APPENDIX A:  REVISION HISTORY

Revision A (6/2017)
Initial release of the document.

Revision B (12/2017)
Standard operating conditions updated in Section 44.0, Electrical Specifications. Other minor corrections.
THE MICROCHIP WEBSITE

Microchip provides online support via our website at 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. 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://www.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.

<table>
<thead>
<tr>
<th>PART NO.</th>
<th>Device</th>
<th>Tape and Reel Option</th>
<th>Temperature Range</th>
<th>Package</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PIC18F26K42, PIC18LF26K42, PIC18F27K42, PIC18LF27K42, PIC18F45K42, PIC18LF45K42, PIC18F46K42, PIC18LF46K42, PIC18F47K42, PIC18LF47K42, PIC18F55K42, PIC18LF55K42, PIC18F56K42, PIC18LF56K42, PIC18F57K42, PIC18LF57K42</td>
<td>Blank = standard packaging (tube or tray) T = Tape and Reel (1), (2)</td>
<td>E = -40°C to +125°C (Extended) I = -40°C to +85°C (Industrial)</td>
<td>ML = 28-lead QFN 6x6mm ML = 44-lead QFN 8x8x0.9mm MX = 28-lead UQFN 6x6x0.5mm MV = 40-lead UQFN 5x5x0.5mm MV = 48-lead UQFN PT = 44-lead TQFP (Thin Quad Flatpack) PT = 48-lead TQFP SO = 28-lead SOIC SP = 28-lead Skinny Plastic DIP SS = 28-lead SSOP</td>
<td>QTP, SQTP, Code or Special Requirements (blank otherwise)</td>
</tr>
</tbody>
</table>

**Examples:**

a) PIC18F26K42-E/P 301 = Extended temp., PDIP package, QTP pattern #301.
b) PIC18F45K42-E/SO = Extended temp., SOIC package.
c) PIC18F46K42T-I/ML = Tape and reel, Industrial temp., QFN package.

**Note 1:** Tape and Reel option is available for ML, MV, PT, SO and SS packages with Industrial Temperature Range only.

**Note 2:** 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.
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.

Trademark

The Microchip name and logo, the Microchip logo, AnyRate, AVR, AVR logo, AVR Freaks, BeaconThings, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, Helio, JukeBlox, KEEL0G, KEEL0G logo, Kleer, LANCheck, LINK MD, maxStylus, maxTouch, MediaLB, megaAVR, MOST, MOST logo, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 logo, Prochip Designer, QTouch, RightTouch, SAM-BA, SpyNic, SST, SST logo, SuperFlash, tinyAVR, UNI/O, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

ClockWorks, The Embedded Control Solutions Company, EtherSynch, Hyper Speed Control, HyperLight Load, IntelliMOS, mTouch, Precision Edge, and Quiet-Wire are registered trademarks of Microchip Technology Incorporated in the U.S.A.


SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

Silicon Storage Technology is a registered trademark 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.

© 2016-2017, Microchip Technology Incorporated, All Rights Reserved.
ISBN: 978-1-5224-2491-8
# Worldwide Sales and Service

<table>
<thead>
<tr>
<th>AMERICAS</th>
<th>ASIA/PACIFIC</th>
<th>ASIA/PACIFIC</th>
<th>EUROPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corporate Office</td>
<td>Australia - Sydney</td>
<td>India - Bangalore</td>
<td>Austria - Wels</td>
</tr>
<tr>
<td>2355 West Chandler Blvd. Chandler, AZ 85224-6199</td>
<td>Tel: 61-2-9888-6733</td>
<td>Tel: 91-80-3090-4444</td>
<td>Tel: 43-7242-2244-39</td>
</tr>
<tr>
<td>Tel: 480-792-7200</td>
<td>China - Beijing</td>
<td>India - New Delhi</td>
<td>Fax: 43-7242-2244-393</td>
</tr>
<tr>
<td>Fax: 480-792-7277</td>
<td>China - Chengdu</td>
<td>India - Pune</td>
<td>Denmark - Copenhagen</td>
</tr>
<tr>
<td>Technical Support: <a href="http://www.microchip.com/support">http://www.microchip.com/support</a></td>
<td>China - Chongqing</td>
<td>Tel: 91-20-4121-0141</td>
<td>Tel: 45-4450-2828</td>
</tr>
<tr>
<td></td>
<td>China - Dongguan</td>
<td>Japan - Osaka</td>
<td>Fax: 45-4485-2829</td>
</tr>
<tr>
<td></td>
<td>China - Guangzhou</td>
<td>Japan - Tokyo</td>
<td>Finland - Espoo</td>
</tr>
<tr>
<td></td>
<td>China - Hangzhou</td>
<td>Korea - Daegu</td>
<td>Tel: 358-9-4520-820</td>
</tr>
<tr>
<td></td>
<td>China - Hong Kong SAR</td>
<td>Korea - Seoul</td>
<td>France - Paris</td>
</tr>
<tr>
<td>Tel: 852-2943-5100</td>
<td>Tel: 82-2-554-7200</td>
<td>Tel: 86-21-3326-8000</td>
<td>Tel: 33-1-69-53-63-20</td>
</tr>
<tr>
<td></td>
<td>Malaysia - Kuala Lumpur</td>
<td>Malaysia - Penang</td>
<td>France - Paris</td>
</tr>
<tr>
<td>Tel: 86-20-8755-8029</td>
<td>Malaysia - Penang</td>
<td>Tel: 60-4-227-8870</td>
<td>Tel: 33-1-69-30-90-79</td>
</tr>
<tr>
<td>Tel: 86-571-8792-8115</td>
<td>Philippines - Manila</td>
<td>Singapore</td>
<td>Germany - Garching</td>
</tr>
<tr>
<td>China - Nanjing</td>
<td>Tel: 86-3-634-9065</td>
<td>Tel: 65-6334-8870</td>
<td>Tel: 49-8931-9700</td>
</tr>
<tr>
<td>Tel: 86-25-8473-2460</td>
<td></td>
<td>Taiwan - Hsin Chu</td>
<td>Germany - Haan</td>
</tr>
<tr>
<td>China - Qingdao</td>
<td>Tel: 886-3-577-8366</td>
<td>Tel: 86-6-2334-2829</td>
<td>Tel: 49-2129-3766400</td>
</tr>
<tr>
<td>Tel: 86-532-8502-7355</td>
<td>Taiwan - Kaohsiung</td>
<td>Taiwan - Taipei</td>
<td>Germany - Heilbronn</td>
</tr>
<tr>
<td>China - Shanghai</td>
<td>Tel: 886-7-213-7830</td>
<td>Tel: 886-2-2508-8600</td>
<td>Tel: 49-7131-67-3636</td>
</tr>
<tr>
<td>Tel: 86-21-3326-8000</td>
<td>Taiwan - Taichung</td>
<td>Thailand - Bangkok</td>
<td>Germany - Karlsruhe</td>
</tr>
<tr>
<td>China - Shenyang</td>
<td>Tel: 886-2-2569-2530</td>
<td>Tel: 66-2-694-1351</td>
<td>Tel: 49-721-625370</td>
</tr>
<tr>
<td>Tel: 86-24-2334-2829</td>
<td>China - Suzhou</td>
<td>Vietnam - Ho Chi Minh</td>
<td>Germany - Munich</td>
</tr>
<tr>
<td>China - Shenzhen</td>
<td>Tel: 86-186-6233-1526</td>
<td>Tel: 84-28-5448-2100</td>
<td>Tel: 49-89-627-144-0</td>
</tr>
<tr>
<td>Tel: 86-755-8864-2200</td>
<td>China - Suzhou</td>
<td></td>
<td>Fax: 49-89-627-144-44</td>
</tr>
<tr>
<td>China - Suzhou</td>
<td>Tel: 86-27-5980-5300</td>
<td></td>
<td>Germany - Rosenheim</td>
</tr>
<tr>
<td>Tel: 86-186-6233-1526</td>
<td>China - Wuhan</td>
<td>Tel: 49-8031-354-560</td>
<td>Tel: 49-8031-354-560</td>
</tr>
<tr>
<td>China - Xinan</td>
<td>Tel: 86-29-8833-7252</td>
<td></td>
<td>Israel - Ra’anana</td>
</tr>
<tr>
<td>Tel: 86-592-2368138</td>
<td>China - Xinan</td>
<td>Tel: 972-9-744-7705</td>
<td>Tel: 972-9-744-7705</td>
</tr>
<tr>
<td>China - Zuhai</td>
<td>Tel: 86-756-3210040</td>
<td></td>
<td>Italy - Milan</td>
</tr>
<tr>
<td>Tel: 86-756-3210040</td>
<td>China - Zuhai</td>
<td>Tel: 39-0331-742611</td>
<td>Tel: 39-0331-742611</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fax: 39-0331-466781</td>
<td>Fax: 39-0331-466781</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Italy - Padova</td>
<td>Tel: 39-049-7625286</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tel: 39-049-7625286</td>
<td>Netherlands - Drunen</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tel: 31-416-690399</td>
<td>Tel: 31-416-690399</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fax: 31-416-690340</td>
<td>Fax: 31-416-690340</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Norway - Trondheim</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tel: 47-7289-7561</td>
<td>Tel: 47-7289-7561</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Poland - Warsaw</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tel: 48-22-3325737</td>
<td>Tel: 48-22-3325737</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Romania - Bucharest</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tel: 40-21-407-87-50</td>
<td>Tel: 40-21-407-87-50</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Spain - Madrid</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tel: 34-91-708-08-90</td>
<td>Tel: 34-91-708-08-90</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fax: 34-91-708-08-91</td>
<td>Fax: 34-91-708-08-91</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Sweden - Gothenberg</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tel: 46-31-704-60-40</td>
<td>Tel: 46-31-704-60-40</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Sweden - Stockholm</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tel: 46-8-5090-4654</td>
<td>Tel: 46-8-5090-4654</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>UK - Wokingham</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tel: 44-118-921-5800</td>
<td>Tel: 44-118-921-5800</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fax: 44-118-921-5820</td>
<td>Fax: 44-118-921-5820</td>
</tr>
</tbody>
</table>