

Welcome to [E-XFL.COM](https://www.e-xfl.com)

### What is "[Embedded - Microcontrollers](#)"?

"[Embedded - Microcontrollers](#)" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

### Applications of "[Embedded - Microcontrollers](#)"

#### Details

|                            |   |
|----------------------------|---|
| Product Status             | Active  |
| Core Processor             | PIC   |
| Core Size                  | 8-Bit   |
| Speed                      | 64MHz   |
| Connectivity               | I <sup>2</sup> C, LINbus, SPI, UART/USART   |
| Peripherals                | Brown-out Detect/Reset, POR, PWM, WDT   |
| Number of I/O              | 60  |
| Program Memory Size        | 64KB (32K x 16)   |
| Program Memory Type        | FLASH   |
| EEPROM Size                | 1K x 8  |
| RAM Size                   | 3.5K x 8  |
| Voltage - Supply (Vcc/Vdd) | 2.3V ~ 5.5V   |
| Data Converters            | A/D 45x10b; D/A 1x5b  |
| Oscillator Type            | Internal  |
| Operating Temperature      | -40°C ~ 125°C (TA)  |
| Mounting Type              | Surface Mount   |
| Package / Case             | 64-VFQFN Exposed Pad  |
| Supplier Device Package    | 64-VQFN (9x9)   |
| Purchase URL               | <a href="https://www.e-xfl.com/product-detail/microchip-technology/pic18f66k40-e-mr">https://www.e-xfl.com/product-detail/microchip-technology/pic18f66k40-e-mr</a> |

# PIC18(L)F65/66K40

## PIC18(L)F6xK40 Family Types

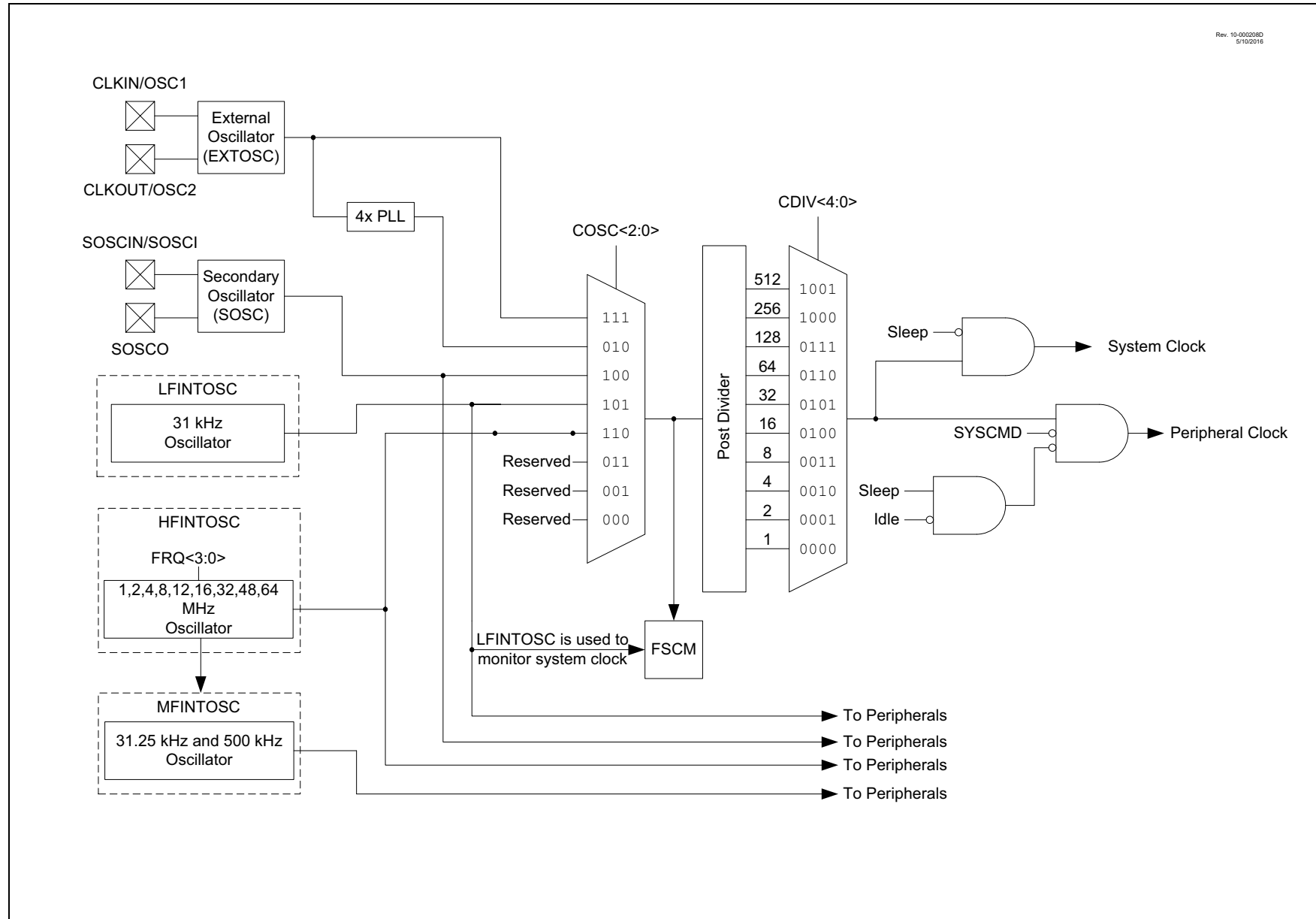
| Device         | Data Sheet Index | Program Memory Flash<br>(words) | Data SRAM<br>(bytes) | Data EEPROM<br>(bytes) | I/O Pins | 16-bit Timers | Comparators | 10-bit ADC <sup>2</sup> with<br>Computation (ch) | 5-bit DAC | Zero-Cross Detect | CCP/10-bit PWM | CWG | Signal Measurement<br>Timer (SMT) | 8-bit TMR with HLT | Windowed Watchdog<br>Timer | CRC with Memory Scan | EUSART | I <sup>2</sup> C/SPI | PPS | Peripheral Module Disable | Temperature Indicator | Debug <sup>(1)</sup> |
|----------------|------------------|---------------------------------|----------------------|------------------------|----------|---------------|-------------|--|-----------|-------------------|----------------|-----|-----------------------------------|--------------------|----------------------------|----------------------|--------|----------------------|-----|---------------------------|-----------------------|----------------------|
| PIC18(L)F65K40 | (1)              | 32k                             | 2048                 | 1024                   | 60       | 5             | 3           | 45   | 1         | 1                 | 5/2            | 1   | 2                                 | 4                  | Y                          | Y                    | 5      | 2                    | Y   | Y                         | Y                     | I                    |
| PIC18(L)F66K40 | (1)              | 64k                             | 3562                 | 1024                   | 60       | 5             | 3           | 45   | 1         | 1                 | 5/2            | 1   | 2                                 | 4                  | Y                          | Y                    | 5      | 2                    | Y   | Y                         | Y                     | I                    |
| PIC18(L)F67K40 | (2)              | 128k                            | 3562                 | 1024                   | 60       | 5             | 3           | 47   | 1         | 1                 | 5/2            | 1   | 2                                 | 4                  | Y                          | Y                    | 5      | 2                    | Y   | Y                         | Y                     | I                    |

**Note 1:** Debugging Methods: (I) – Integrated on Chip.

**Data Sheet Index:** (Unshaded devices are described in this document.)

- DS40001842 PIC18(L)F65/66K40 Data Sheet, 64-Pin, 8-bit Flash Microcontrollers
- DS40001841 PIC18(L)F67K40 Data Sheet, 64-Pin, 8-bit Flash Microcontrollers

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

**FIGURE 4-1: SIMPLIFIED PIC® MCU CLOCK SOURCE BLOCK DIAGRAM**

## 4.3.2.1 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 4.4 "Clock Switching"** for more information.

The HFINTOSC frequency can be selected by setting the HFFRQ<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.

## 4.3.2.2 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.

## 4.3.2.3 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 4-3).

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.

## 4.3.2.4 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 4.4, Clock Switching** for more information.

## 4.3.2.5 ADCRC

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

# PIC18(L)F65/66K40

**TABLE 10-5: REGISTER FILE SUMMARY FOR PIC18(L)F65K40 DEVICES (CONTINUED)**

| Address | Name       | Bit 7  | Bit 6     | Bit 5     | Bit 4       | Bit 3      | Bit 2         | Bit 1    | Bit 0 | Value on<br>POR, BOR |
|---------|------------|--|-----------|-----------|-------------|------------|---------------|----------|-------|----------------------|
| F42h    | CRCACCH    | ACC<15:8>                                    |           |           |             |            |               |          |       | 00000000             |
| F41h    | CRCACCL    | ACC<7:0>                                     |           |           |             |            |               |          |       | 00000000             |
| F40h    | CRCDATH    | DATA<15:8>                                   |           |           |             |            |               |          |       | xxxxxxxx             |
| F3Fh    | CRCDATL    | DATA<7:0>                                    |           |           |             |            |               |          |       | xxxxxxxx             |
| F3Eh    | CWG1STR    | OVRD   | OVRC      | OVRB      | OVRA        | STRD       | STRC          | STRB     | STRA  | 00000000             |
| F3Dh    | CWG1AS1    | AS7E   | AS6E      | AS5E      | AS4E        | AS3E       | AS2E          | AS1E     | AS0E  | 00000000             |
| F3Ch    | CWG1AS0    | SHUTDOWN                                     | REN       | LSBD<1:0> |             | LSAC<1:0>  |               | —        | —     | 000101--             |
| F3Bh    | CWG1CON1   | —  | —         | IN        | —           | POLD       | POLC          | POLB     | POLA  | --x-0000             |
| F3Ah    | CWG1CON0   | EN   | LD        | —         | —           | —          | MODE<2:0>     |          |       | 00---000             |
| F39h    | CWG1DBF    | —  | —         | DBF<5:0>  |             |            |               |          |       | --000000             |
| F38h    | CWG1DBR    | —  | —         | DBR<5:0>  |             |            |               |          |       | --000000             |
| F37h    | CWG1ISM    | —  | —         | —         | —           | ISM<3:0>   |               |          |       | ----0000             |
| F36h    | CWG1CLKCON | —  | —         | —         | —           | —          | —             | —        | CS    | -----0               |
| F35h    | CLKRCLK    | —  | —         | —         | —           | —          | CLKRxCLK<2:0> |          |       | -----000             |
| F34h    | CLKRCON    | EN   | —         | —         | CLKRDC<1:0> |            | CLKRDIV<2:0>  |          |       | 0--10000             |
| F33h    | T7CLK      | —  | —         | —         | —           | CS<3:0>    |               |          |       | ----0000             |
| F32h    | T7GATE     | —  | —         | —         | —           | GSS<3:0>   |               |          |       | ----0000             |
| F31h    | T7GCON     | GE   | GPOL      | GTM       | GSPM        | GO/DONE    | GVAL          | —        | —     | 00000x--             |
| F30h    | T7CON      | —  | —         | CKPS<1:0> |             | —          | SYNC          | RD16     | ON    | --00-000             |
| F2Fh    | TMR7H      | TMR5H<7:0>                                   |           |           |             |            |               |          |       | 00000000             |
| F2Eh    | TMR7L      | TMR5L<7:0>                                   |           |           |             |            |               |          |       | 00000000             |
| F2Dh    | T8RST      | —  | —         | —         | —           | RSEL<3:0>  |               |          |       | ----0000             |
| F2Ch    | T8CLKCON   | —  | —         | —         | —           | CS<3:0>    |               |          |       | ----0000             |
| F2Bh    | T8HLT      | PSYNC  | CPOL      | CSYNC     | MODE<4:0>   |            |               |          |       | 00000000             |
| F2Ah    | T8CON      | ON   | CKPS<2:0> |           |             | OUTPS<3:0> |               |          |       | 00000000             |
| F29h    | T8PR       | TMR2 Period Register                         |           |           |             |            |               |          |       | 11111111             |
| F28h    | T8TMR      | Holding Register for the 8-bit TMR2 Register |           |           |             |            |               |          |       | 00000000             |
| F27h    | CCP3CAP    | —  | —         | —         | —           | —          | —             | CTS<1:0> |       | -----00              |
| F26h    | CCP3CON    | EN   | —         | OUT       | FMT         | MODE<3:0>  |               |          |       | 0-000000             |
| F25h    | CCP3H      | Capture/Compare/PWM Register 3 (MSB)         |           |           |             |            |               |          |       | xxxxxxxx             |
| F24h    | CCP3L      | Capture/Compare/PWM Register 3 (LSB)         |           |           |             |            |               |          |       | xxxxxxxx             |
| F23h    | CCP4CAP    | —  | —         | —         | —           | —          | —             | CTS<1:0> |       | -----00              |
| F22h    | CCP4CON    | EN   | —         | OUT       | FMT         | MODE<3:0>  |               |          |       | 0-000000             |
| F21h    | CCPR4H     | Capture/Compare/PWM Register 4 (MSB)         |           |           |             |            |               |          |       | xxxxxxxx             |
| F20h    | CCPR4L     | Capture/Compare/PWM Register 4 (LSB)         |           |           |             |            |               |          |       | xxxxxxxx             |
| F1Fh    | CCP5CAP    | —  | —         | —         | —           | —          | —             | CTS<1:0> |       | -----00              |
| F1Eh    | CCP5CON    | EN   | —         | OUT       | FMT         | MODE<3:0>  |               |          |       | 0-000000             |
| F1Dh    | CCPR5H     | Capture/Compare/PWM Register 5 (MSB)         |           |           |             |            |               |          |       | xxxxxxxx             |
| F1Ch    | CCPR5L     | Capture/Compare/PWM Register 5 (LSB)         |           |           |             |            |               |          |       | xxxxxxxx             |
| F1Bh    | SMT1WIN    | —  | —         | —         | WSEL<4:0>   |            |               |          |       | --00000              |
| F1Ah    | SMT1SIG    | —  | —         | —         | SSEL<4:0>   |            |               |          |       | --00000              |

**Legend:** x = unknown, u = unchanged, — = unimplemented,  $\bar{c}$  = value depends on condition

**Note 1:** Not available on LF devices.

Example 12-3 shows the sequence to do a 16 x 16 unsigned multiplication. Equation 12-1 shows the algorithm that is used. The 32-bit result is stored in four registers (RES<3:0>).

## EQUATION 12-1: 16 x 16 UNSIGNED MULTIPLICATION ALGORITHM

$$\begin{aligned} \text{RES3:RES0} &= \text{ARG1H:ARG1L} \cdot \text{ARG2H:ARG2L} \\ &= (\text{ARG1H} \cdot \text{ARG2H} \cdot 2^{16}) + \\ &\quad (\text{ARG1H} \cdot \text{ARG2L} \cdot 2^8) + \\ &\quad (\text{ARG1L} \cdot \text{ARG2H} \cdot 2^8) + \\ &\quad (\text{ARG1L} \cdot \text{ARG2L}) \end{aligned}$$

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

```

MOVWF ARG1L, W           ; ARG1L * ARG2L->
MULWF ARG2L               ; PRODH:PRODL

MOVFF PRODH, RES1         ;
MOVFF PRODL, RES0         ;
;
MOVWF ARG1H, W           ; ARG1H * ARG2H->
MULWF ARG2H               ; PRODH:PRODL

MOVFF PRODH, RES3         ;
MOVFF PRODL, RES2         ;
;
MOVWF ARG1L, W           ; ARG1L * ARG2H->
MULWF ARG2H               ; PRODH:PRODL

MOVWF PRODL, W           ;
ADDWF RES1, F             ; Add cross
MOVWF PRODH, W           ; products
ADDWFC RES2, F            ;
CLRF WREG                 ;
ADDWFC RES3, F            ;
;
MOVWF ARG1H, W           ;
MULWF ARG2L               ; ARG1H * ARG2L->
                        ; PRODH:PRODL

MOVWF PRODL, W           ;
ADDWF RES1, F             ; Add cross
MOVWF PRODH, W           ; products
ADDWFC RES2, F            ;
CLRF WREG                 ;
ADDWFC RES3, F            ;

```

Example 12-4 shows the sequence to do a 16 x 16 signed multiply. Equation 12-2 shows the algorithm used. The 32-bit result is stored in four registers (RES<3:0>). To account for the sign bits of the arguments, the MSb for each argument pair is tested and the appropriate subtractions are done.

## EQUATION 12-2: 16 x 16 SIGNED MULTIPLICATION ALGORITHM

$$\begin{aligned} \text{RES3:RES0} &= \text{ARG1H:ARG1L} \cdot \text{ARG2H:ARG2L} \\ &= (\text{ARG1H} \cdot \text{ARG2H} \cdot 2^{16}) + \\ &\quad (\text{ARG1H} \cdot \text{ARG2L} \cdot 2^8) + \\ &\quad (\text{ARG1L} \cdot \text{ARG2H} \cdot 2^8) + \\ &\quad (\text{ARG1L} \cdot \text{ARG2L}) + \\ &\quad (-1 \cdot \text{ARG2H} < 7 > \cdot \text{ARG1H:ARG1L} \cdot 2^{16}) + \\ &\quad (-1 \cdot \text{ARG1H} < 7 > \cdot \text{ARG2H:ARG2L} \cdot 2^{16}) \end{aligned}$$

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

```

MOVWF ARG1L, W           ; ARG1L * ARG2L ->
MULWF ARG2L               ; PRODH:PRODL

MOVFF PRODH, RES1         ;
MOVFF PRODL, RES0         ;
;
MOVWF ARG1H, W           ; ARG1H * ARG2H ->
MULWF ARG2H               ; PRODH:PRODL

MOVFF PRODH, RES3         ;
MOVFF PRODL, RES2         ;
;
MOVWF ARG1L, W           ; ARG1L * ARG2H ->
MULWF ARG2H               ; PRODH:PRODL

MOVWF PRODL, W           ;
ADDWF RES1, F             ; Add cross
MOVWF PRODH, W           ; products
ADDWFC RES2, F            ;
CLRF WREG                 ;
ADDWFC RES3, F            ;
;
MOVWF ARG1H, W           ;
MULWF ARG2L               ; ARG1H * ARG2L ->
                        ; PRODH:PRODL

MOVWF PRODL, W           ;
ADDWF RES1, F             ; Add cross
MOVWF PRODH, W           ; products
ADDWFC RES2, F            ;
CLRF WREG                 ;
ADDWFC RES3, F            ;
;
BTFSS ARG2H, 7            ; ARG2H:ARG2L neg?
BRA SIGN_ARG1             ; no, check ARG1
MOVWF ARG1L, W           ;
SUBWF RES2                ;
MOVWF ARG1H, W           ;
SUBWFB RES3               ;
;
SIGN_ARG1
BTFSS ARG1H, 7            ; ARG1H:ARG1L neg?
BRA CONT_CODE             ; no, done
MOVWF ARG2L, W           ;
SUBWF RES2                ;
MOVWF ARG2H, W           ;
SUBWFB RES3               ;
;
CONT_CODE
:
```

## 13.9 Program Memory Scan Configuration

If desired, the program memory scan module may be used in conjunction with the CRC module to perform a CRC calculation over a range of program memory addresses. In order to set up the scanner to work with the CRC you need to perform the following steps:

1. Set the Enable bit in both the CRCCON0 and SCANCON0 registers. If they get disabled, all internal states of the scanner and the CRC are reset (registers are unaffected).
2. Choose which memory access mode is to be used (see **Section 13.11 “Scanning Modes”**) and set the MODE bits of the SCANCON0 register appropriately.
3. Based on the memory access mode, set the INTM bits of the SCANCON0 register to the appropriate interrupt mode (see **Section 13.11.5 “Interrupt Interaction”**).
4. Set the SCANLADRL/H/U and SCANHADRL/H/U registers with the beginning and ending locations in memory that are to be scanned.
5. The CRCGO bit must be set before setting the SCANGO bit. Setting the SCANGO bit starts the scan. Both CRCEN and CRCGO bits must be enabled to use the scanner. When either of these bits are disabled, the scan aborts and the INVALID bit SCANCON0 is set. The scanner will wait for the signal from the CRC that it is ready for the first Flash memory location, then begin loading data into the CRC. It will continue to do so until it either hits the configured end address or an address that is unimplemented on the device, at which point the SCANGO bit will clear, Scanner functions will cease, and the SCANIF interrupt will be triggered. Alternately, the SCANGO bit can be cleared in software if desired.

## 13.10 Scanner Interrupt

The scanner will trigger an interrupt when the SCANGO bit transitions from ‘1’ to ‘0’. The SCANIF interrupt flag of PIR7 is set when the last memory location is reached and the data is entered into the CRCDATA registers. The SCANIF bit can only be cleared in software. The SCAN interrupt enable is the SCANIE bit of the PIE7 register.

## 13.11 Scanning Modes

The memory scanner can scan in four modes: Burst, Peek, Concurrent, and Triggered. These modes are controlled by the MODE bits of the SCANCON0 register. The four modes are summarized in Table 13-2.

### 13.11.1 BURST MODE

When MODE = 01, the scanner is in Burst mode. In Burst mode, CPU operation is stalled beginning with the operation after the one that sets the SCANGO bit, and the scan begins, using the instruction clock to execute. The CPU is held in its current state until the scan stops. Note that because the CPU is not executing instructions, the SCANGO bit cannot be cleared in software, so the CPU will remain stalled until one of the hardware end-conditions occurs. Burst mode has the highest throughput for the scanner, but has the cost of stalling other execution while it occurs.

### 13.11.2 CONCURRENT MODE

When MODE = 00, the scanner is in Concurrent mode. Concurrent mode, like Burst mode, stalls the CPU while performing accesses of memory. However, while Burst mode stalls until all accesses are complete, Concurrent mode allows the CPU to execute in between access cycles.

### 13.11.3 TRIGGERED MODE

When MODE = 11, the scanner is in Triggered mode. Triggered mode behaves identically to Concurrent mode, except instead of beginning the scan immediately upon the SCANGO bit being set, it waits for a rising edge from a separate trigger clock, the source of which is determined by the SCANTRIG register.

### 13.11.4 PEEK MODE

When MODE = 10, the scanner is in Peek mode. Peek mode waits for an instruction cycle in which the CPU does not need to access the NVM (such as a branch instruction) and uses that cycle to do its own NVM access. This results in the lowest throughput for the NVM access (and can take a much longer time to complete a scan than the other modes), but does so without any impact on execution times, unlike the other modes.

## 14.9 INTn Pin Interrupts

PIC18(L)F6xK40 devices have four external interrupt sources which can be assigned to any pin on PORTA and PORTB using PPS. 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.

All external interrupts (INT0, INT1, INT2 and INT3) can wake-up the processor from Idle or Sleep modes if bit INTxE was set prior to going into those modes. If the Global Interrupt Enable bit, GIE/GIEH, is set, the processor will branch to the interrupt vector following wake-up.

Interrupt priority is determined by the value contained in the interrupt priority bits, INT0IP, INT1IP, INT2IP and INT3 of the IPR0 register.

## 14.10 TMR0 Interrupt

In 8-bit mode (which is the default), an overflow in the TMR0 register (FFh → 00h) will set flag bit, TMR0IF. In 16-bit mode, an overflow in the TMR0H:TMR0L register pair (FFFFh → 0000h) will set TMR0IF. The interrupt can be enabled/disabled by setting/clearing enable bit, TMR0IE of the PIE0 register. Interrupt priority for Timer0 is determined by the value contained in the interrupt priority bit, TMR0IP of the IPR0 register. See **Section 18.0 “Timer0 Module”** for further details on the Timer0 module.

## 14.11 Interrupt-on-Change

An input change on any port pins that support IOC sets Flag bit, IOCIF of the PIR0 register. The interrupt can be enabled/disabled by setting/clearing the enable bit, IOCIE of the PIE0 register. Pins must also be individually enabled in the IOCxP and IOCxN register. IOCIF is a read-only bit and the flag can be cleared by clearing the corresponding IOCxF registers. For more information refer to **Section 16.0 “Interrupt-on-Change”**.

## 14.12 Context Saving During Interrupts

During interrupts, the return PC address is saved on the stack. Additionally, the WREG, STATUS and BSR registers are saved on the fast return stack. If a fast return from interrupt is not used (see **Section 10.2.2 “Fast Register Stack”**), the user may need to save the WREG, STATUS and BSR registers on entry to the Interrupt Service Routine. Depending on the user's application, other registers may also need to be saved. Example 14-1 saves and restores the WREG, STATUS and BSR registers during an Interrupt Service Routine.

### EXAMPLE 14-1: SAVING STATUS, WREG AND BSR REGISTERS IN RAM

```
MOVWF    W_TEMP                ; W_TEMP is in virtual bank
MOVFF    STATUS, STATUS_TEMP    ; STATUS_TEMP located anywhere
MOVFF    BSR, BSR_TEMP          ; BSR_TEMP located anywhere
;
; USER ISR CODE
;
MOVFF    BSR_TEMP, BSR          ; Restore BSR
MOVF     W_TEMP, W              ; Restore WREG
MOVFF    STATUS_TEMP, STATUS    ; Restore STATUS
```

## 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 EUSART1 RX input, set RXxPPS to 6'b01 0111, and to select PORTC<6> as the EUSART1 TX output set RC6PPS to 6'b00 1100.

### 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:

- EUSART (synchronous operation)
- MSSP (I<sup>2</sup>C)

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.

## REGISTER 17-2: RxyPPS: PIN Rxy OUTPUT SOURCE SELECTION REGISTER

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

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

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

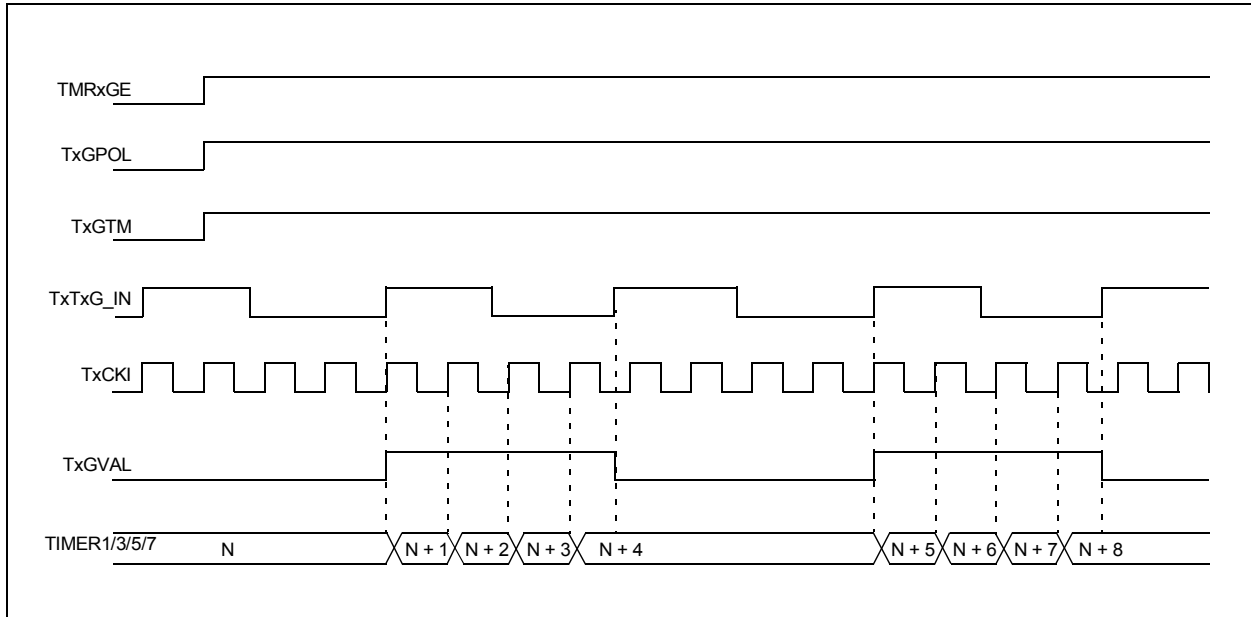
'1' = Bit is set

'0' = Bit is cleared

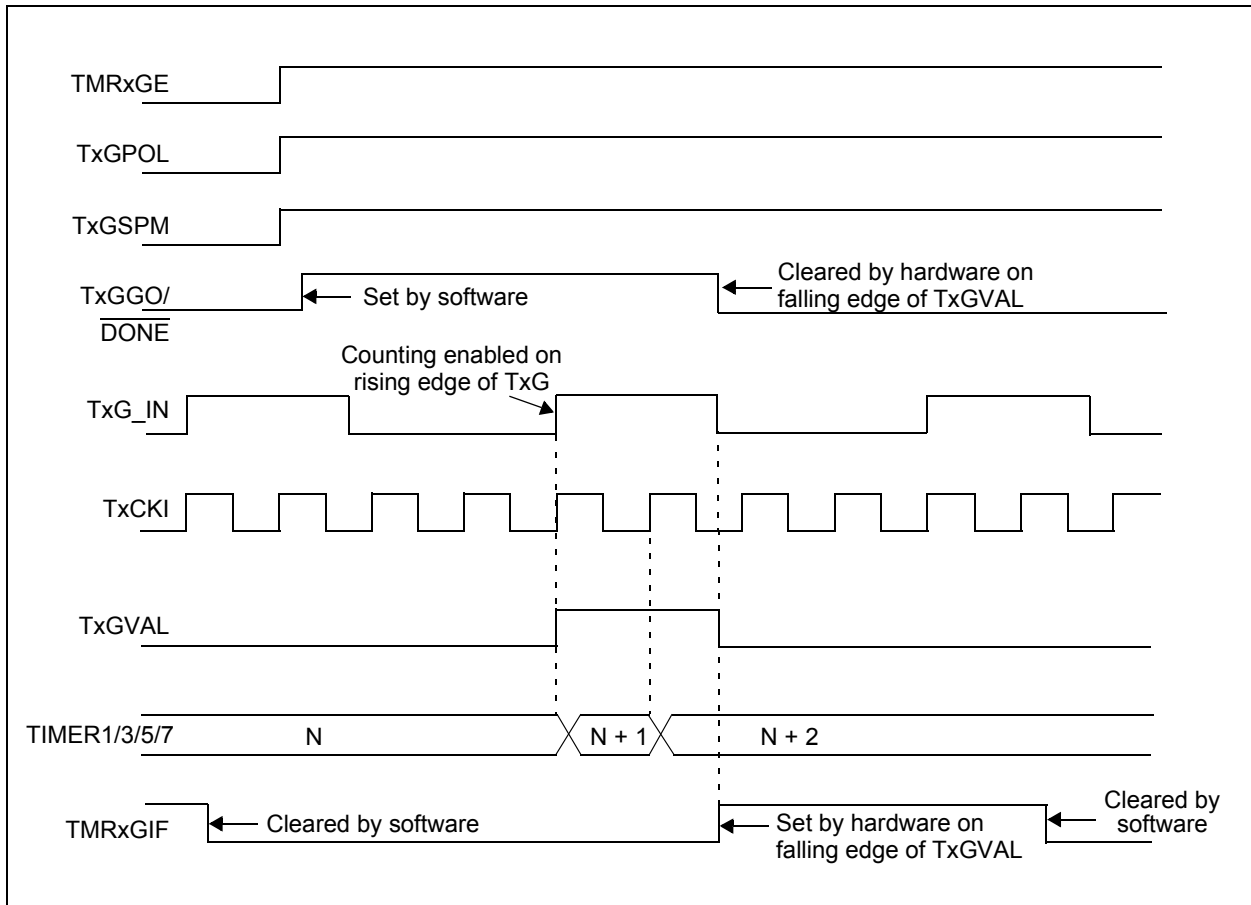
bit 7-6

**Unimplemented:** Read as '0'

**FIGURE 19-5: TIMER1/3/5/7 GATE TOGGLE MODE**



**FIGURE 19-6: TIMER1/3/5/7 GATE SINGLE-PULSE MODE**



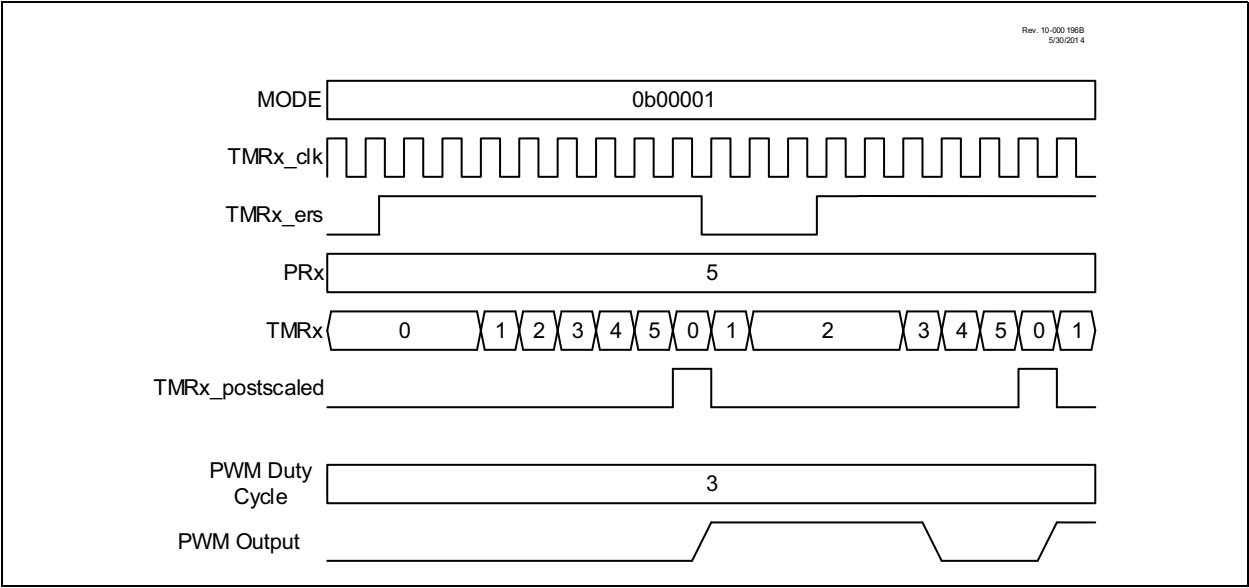
20.5.2 HARDWARE GATE MODE

The Hardware Gate modes operate the same as the Software Gate mode except the TMRx\_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 20-5 illustrates the Hardware Gating mode for MODE<4:0> = 00001 in which a high input level starts the counter.

FIGURE 20-5: HARDWARE GATE MODE TIMING DIAGRAM (MODE = 00001)



## REGISTER 21-2: CCPTMRS0: CCP TIMERS CONTROL REGISTER 0

| R/W-0/0     | R/W-1/1 | R/W-0/0     | R/W-1/1 | R/W-0/0     | R/W-1/1 | R/W-0/0     | R/W-1/1 |
|-------------|---------|-------------|---------|-------------|---------|-------------|---------|
| C4TSEL<1:0> |         | C3TSEL<1:0> |         | C2TSEL<1:0> |         | C1TSEL<1:0> |         |
| bit 7       |         |             |         |             |         |             | bit 0   |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-6

**C4TSEL<1:0>**: CCP4 Timer Selection bits

11 = CCP4 is based off Timer7 in Capture/Compare mode and Timer8 in PWM mode

10 = CCP4 is based off Timer5 in Capture/Compare mode and Timer6 in PWM mode

01 = CCP4 is based off Timer3 in Capture/Compare mode and Timer4 in PWM mode

00 = CCP4 is based off Timer1 in Capture/Compare mode and Timer2 in PWM mode

bit 5-4

**C3TSEL<1:0>**: CCP3 Timer Selection bits

11 = CCP3 is based off Timer7 in Capture/Compare mode and Timer8 in PWM mode

10 = CCP3 is based off Timer5 in Capture/Compare mode and Timer6 in PWM mode

01 = CCP3 is based off Timer3 in Capture/Compare mode and Timer4 in PWM mode

00 = CCP3 is based off Timer1 in Capture/Compare mode and Timer2 in PWM mode

bit 3-2

**C2TSEL<1:0>**: CCP2 Timer Selection bits

11 = CCP2 is based off Timer7 in Capture/Compare mode and Timer8 in PWM mode

10 = CCP2 is based off Timer5 in Capture/Compare mode and Timer6 in PWM mode

01 = CCP2 is based off Timer3 in Capture/Compare mode and Timer4 in PWM mode

00 = CCP2 is based off Timer1 in Capture/Compare mode and Timer2 in PWM mode

bit 1-0

**C1TSEL<1:0>**: CCP1 Timer Selection bits

11 = CCP1 is based off Timer7 in Capture/Compare mode and Timer8 in PWM mode

10 = CCP1 is based off Timer5 in Capture/Compare mode and Timer6 in PWM mode

01 = CCP1 is based off Timer3 in Capture/Compare mode and Timer4 in PWM mode

00 = CCP1 is based off Timer1 in Capture/Compare mode and Timer2 in PWM mode

## 22.2 Register Definitions: PWM Control

Long bit name prefixes for the PWM peripherals are shown in Table 22-3. Refer to **Section 1.4.2.2 “Long Bit Names”** for more information.

**TABLE 22-3:**

| Peripheral | Bit Name Prefix |
|------------|-----------------|
| PWM6       | PWM6            |
| PWM7       | PWM7            |

**REGISTER 22-1: PWMxCON: PWM CONTROL REGISTER**

|         |     |       |         |     |     |     |       |
|---------|-----|-------|---------|-----|-----|-----|-------|
| R/W-0/0 | U-0 | R-0/0 | R/W-0/0 | U-0 | U-0 | U-0 | U-0   |
| EN      | —   | OUT   | POL     | —   | —   | —   | —     |
| bit 7   |     |       |         |     |     |     | bit 0 |

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as ‘0’

u = Bit is unchanged

x = Bit is unknown

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

‘1’ = Bit is set

‘0’ = Bit is cleared

- bit 7      **EN:** PWM Module Enable bit  
             1 = PWM module is enabled  
             0 = PWM module is disabled
- bit 6      **Unimplemented:** Read as ‘0’
- bit 5      **OUT:** PWM Module Output Level When Bit is Read
- bit 4      **POL:** PWM Output Polarity Select bit  
             1 = PWM output is inverted  
             0 = PWM output is normal
- bit 3-0    **Unimplemented:** Read as ‘0’

## 24.12 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 the system clock and the CWG clock source, 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.

## 24.13 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 CWG1CON0 register to set the output operating mode.
4. Configure the POLy bits of the CWG1CON1 register to set the output polarities.
5. Configure the ISM<3:0> bits of the CWG1ISM register to select the data input source.
6. If a steering mode is selected, configure the STRx bits to select the desired output on the CWG outputs.
7. Configure the LSB<1:0> and LSAC<1:0> bits of the CWG1ASD0 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 CWG1AS0.
9. If auto-shutdown is desired, configure the ASxE bits of the CWG1AS1 register to select the shutdown source.
10. Set the desired rising and falling dead-band times with the CWG1DBR and CWG1DBF registers.
11. Select the clock source in the CWG1CLKCON 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.

## REGISTER 28-2: RCxSTA: RECEIVE STATUS AND CONTROL REGISTER

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R-0/0 | R/HC-0/0 | R/HC-0/0 |
|---------|---------|---------|---------|---------|-------|----------|----------|
| SPEN    | RX9     | SREN    | CREN    | ADDEN   | FERR  | OERR     | RX9D     |
| bit 7   |         |         |         |         |       |          | bit 0    |

### Legend:

R = Readable bit

W = Writable bit

HC = Bit is cleared by hardware

u = Bit is unchanged

x = Bit is unknown

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

'1' = Bit is set

'0' = Bit is cleared

|       |   |
|-------|---|
| bit 7 | <b>SPEN:</b> Serial Port Enable bit<br>1 = Serial port enabled<br>0 = Serial port disabled (held in Reset)  |
| bit 6 | <b>RX9:</b> 9-Bit Receive Enable bit<br>1 = Selects 9-bit reception<br>0 = Selects 8-bit reception  |
| bit 5 | <b>SREN:</b> Single Receive Enable bit<br><u>Asynchronous mode:</u><br>Don't care<br><u>Synchronous mode – Master:</u><br>1 = Enables single receive<br>0 = Disables single receive<br>This bit is cleared after reception is complete.<br><u>Synchronous mode – Slave</u><br>Don't care  |
| bit 4 | <b>CREN:</b> Continuous Receive Enable bit<br><u>Asynchronous mode:</u><br>1 = Enables receiver<br>0 = Disables receiver<br><u>Synchronous mode:</u><br>1 = Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN)<br>0 = Disables continuous receive  |
| bit 3 | <b>ADDEN:</b> Address Detect Enable bit<br><u>Asynchronous mode 9-bit (RX9 = 1):</u><br>1 = Enables address detection, enable interrupt and load the receive buffer when RSR<8> is set<br>0 = Disables address detection, all bytes are received and ninth bit can be used as parity bit<br><u>Asynchronous mode 8-bit (RX9 = 0):</u><br>Don't care |
| bit 2 | <b>FERR:</b> Framing Error bit<br>1 = Framing error (can be updated by reading RCxREG register and receive next valid byte)<br>0 = No framing error   |
| bit 1 | <b>OERR:</b> Overrun Error bit<br>1 = Overrun error (can be cleared by clearing bit CREN)<br>0 = No overrun error   |
| bit 0 | <b>RX9D:</b> Ninth bit of Received Data<br>This can be address/data bit or a parity bit and must be calculated by user firmware.  |

## 32.3 ADC Acquisition Requirements

For the ADC to meet its specified accuracy, the charge holding capacitor (CHOLD) must be allowed to fully charge to the input channel voltage level. The Analog Input model is shown in Figure 32-4. The source impedance (RS) and the internal sampling switch (RSS) impedance directly affect the time required to charge the capacitor CHOLD. The sampling switch (RSS) impedance varies over the device voltage (VDD), refer to Figure 32-4. **The maximum recommended impedance for analog sources is 10 kΩ.** As 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 32-1 may be used. This equation assumes that 1/2 LSB error is used (1,024 steps for the ADC). The 1/2 LSB error is the maximum error allowed for the ADC to meet its specified resolution.

### EQUATION 32-1: ACQUISITION TIME EXAMPLE

*Assumptions: Temperature = 50°C and external impedance of 10kΩ 5.0V VDD*

$$\begin{aligned} T_{ACQ} &= \text{Amplifier Settling Time} + \text{Hold Capacitor Charging Time} + \text{Temperature Coefficient} \\ &= T_{AMP} + T_C + T_{COFF} \\ &= 2\mu s + T_C + [(Temperature - 25^\circ C)(0.05\mu s/^\circ C)] \end{aligned}$$

*The value for TC can be approximated with the following equations:*

$$V_{APPLIED} \left( 1 - \frac{1}{(2^{n+1}) - 1} \right) = V_{CHOLD} \quad ;[1] \text{ } V_{CHOLD} \text{ charged to within } 1/2 \text{ lsb}$$

$$V_{APPLIED} \left( 1 - e^{\frac{-T_C}{RC}} \right) = V_{CHOLD} \quad ;[2] \text{ } V_{CHOLD} \text{ charge response to } V_{APPLIED}$$

$$V_{APPLIED} \left( 1 - e^{\frac{-T_C}{RC}} \right) = V_{APPLIED} \left( 1 - \frac{1}{(2^{n+1}) - 1} \right) \quad ;\text{combining [1] and [2]}$$

*Note: Where n = number of bits of the ADC.*

*Solving for TC:*

$$\begin{aligned} T_C &= -CHOLD(RIC + RSS + RS) \ln(1/2047) \\ &= -10pF(1k\Omega + 7k\Omega + 10k\Omega) \ln(0.0004885) \\ &= 1.37\mu s \end{aligned}$$

*Therefore:*

$$\begin{aligned} T_{ACQ} &= 2\mu s + 892ns + [(50^\circ C - 25^\circ C)(0.05\mu s/^\circ C)] \\ &= 4.62\mu s \end{aligned}$$

**Note 1:** The reference voltage (VREF) has no effect on the equation, since it cancels itself out.


**2:** The charge holding capacitor (CHOLD) is not discharged after each conversion.

**3:** The maximum recommended impedance for analog sources is 10 kΩ. This is required to meet the pin leakage specification.

**TABLE 36-2: INSTRUCTION SET**

| Mnemonic,<br>Operands    | Description                     | Cycles  | 16-Bit Instruction Word |      |      |      | Status<br>Affected | Notes           |            |
|--------------------------|---------------------------------|---|-------------------------|------|------|------|--------------------|-----------------|------------|
|                          |                                 |   | MSb                     |      | LSb  |      |                    |                 |            |
| BYTE-ORIENTED OPERATIONS |                                 |   |                         |      |      |      |                    |                 |            |
| ADDWF                    | f, d, a                         | Add WREG and f  | 1                       | 0010 | 01da | ffff | ffff               | C, DC, Z, OV, N | 1, 2       |
| ADDWFC                   | f, d, a                         | Add WREG and CARRY bit to f   | 1                       | 0010 | 00da | ffff | ffff               | C, DC, Z, OV, N | 1, 2       |
| ANDWF                    | f, d, a                         | AND WREG with f   | 1                       | 0001 | 01da | ffff | ffff               | Z, N            | 1, 2       |
| CLRF                     | f, a                            | Clear f   | 1                       | 0110 | 101a | ffff | ffff               | Z               | 2          |
| COMF                     | f, d, a                         | Complement f  | 1                       | 0001 | 11da | ffff | ffff               | Z, N            | 1, 2       |
| CPFSEQ                   | f, a                            | Compare f with WREG, skip =   | 1 (2 or 3)              | 0110 | 001a | ffff | ffff               | None            | 4          |
| CPFSGT                   | f, a                            | Compare f with WREG, skip >   | 1 (2 or 3)              | 0110 | 010a | ffff | ffff               | None            | 4          |
| CPFSLT                   | f, a                            | Compare f with WREG, skip <   | 1 (2 or 3)              | 0110 | 000a | ffff | ffff               | None            | 1, 2       |
| DECf                     | f, d, a                         | Decrement f   | 1                       | 0000 | 01da | ffff | ffff               | C, DC, Z, OV, N | 1, 2, 3, 4 |
| DECFSZ                   | f, d, a                         | Decrement f, Skip if 0  | 1 (2 or 3)              | 0010 | 11da | ffff | ffff               | None            | 1, 2, 3, 4 |
| DCFSNZ                   | f, d, a                         | Decrement f, Skip if Not 0  | 1 (2 or 3)              | 0100 | 11da | ffff | ffff               | None            | 1, 2       |
| INCF                     | f, d, a                         | Increment f   | 1                       | 0010 | 10da | ffff | ffff               | C, DC, Z, OV, N | 1, 2, 3, 4 |
| INCFSZ                   | f, d, a                         | Increment f, Skip if 0  | 1 (2 or 3)              | 0011 | 11da | ffff | ffff               | None            | 4          |
| INFSNZ                   | f, d, a                         | Increment f, Skip if Not 0  | 1 (2 or 3)              | 0100 | 10da | ffff | ffff               | None            | 1, 2       |
| IORWF                    | f, d, a                         | Inclusive OR WREG with f  | 1                       | 0001 | 00da | ffff | ffff               | Z, N            | 1, 2       |
| MOVF                     | f, d, a                         | Move f  | 1                       | 0101 | 00da | ffff | ffff               | Z, N            | 1          |
| MOVFF                    | f <sub>s</sub> , f <sub>d</sub> | Move f <sub>s</sub> (source) to 1st word<br>f <sub>d</sub> (destination) 2nd word | 2                       | 1100 | ffff | ffff | ffff               | None            |            |
|                          |                                 |   |                         | 1111 | ffff | ffff | ffff               |                 |            |
| MOVWF                    | f, a                            | Move WREG to f  | 1                       | 0110 | 111a | ffff | ffff               | None            |            |
| MULWF                    | f, a                            | Multiply WREG with f  | 1                       | 0000 | 001a | ffff | ffff               | None            | 1, 2       |
| NEGF                     | f, a                            | Negate f  | 1                       | 0110 | 110a | ffff | ffff               | C, DC, Z, OV, N |            |
| RLCF                     | f, d, a                         | Rotate Left f through Carry   | 1                       | 0011 | 01da | ffff | ffff               | C, Z, N         | 1, 2       |
| RLNCF                    | f, d, a                         | Rotate Left f (No Carry)  | 1                       | 0100 | 01da | ffff | ffff               | Z, N            |            |
| RRCF                     | f, d, a                         | Rotate Right f through Carry  | 1                       | 0011 | 00da | ffff | ffff               | C, Z, N         |            |
| RRNCF                    | f, d, a                         | Rotate Right f (No Carry)   | 1                       | 0100 | 00da | ffff | ffff               | Z, N            |            |
| SETF                     | f, a                            | Set f   | 1                       | 0110 | 100a | ffff | ffff               | None            | 1, 2       |
| SUBFWB                   | f, d, a                         | Subtract f from WREG with<br>borrow   | 1                       | 0101 | 01da | ffff | ffff               | C, DC, Z, OV, N |            |
| SUBWF                    | f, d, a                         | Subtract WREG from f  | 1                       | 0101 | 11da | ffff | ffff               | C, DC, Z, OV, N | 1, 2       |
| SUBWFB                   | f, d, a                         | Subtract WREG from f with<br>borrow   | 1                       | 0101 | 10da | ffff | ffff               | C, DC, Z, OV, N |            |
| SWAPF                    | f, d, a                         | Swap nibbles in f   | 1                       | 0011 | 10da | ffff | ffff               | None            | 4          |
| TSTFSZ                   | f, a                            | Test f, skip if 0   | 1 (2 or 3)              | 0110 | 011a | ffff | ffff               | None            | 1, 2       |
| XORWF                    | f, d, a                         | Exclusive OR WREG with f  | 1                       | 0001 | 10da | ffff | ffff               | Z, N            |            |

- Note 1:** When a PORT register is modified as a function of itself (e.g., `MOVF PORTB, 1, 0`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.
- 2:** If this instruction is executed on the TMR0 register (and where applicable, 'd' = 1), the prescaler will be cleared if assigned.
- 3:** If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a `NOP`.
- 4:** Some instructions are two-word instructions. The second word of these instructions will be executed 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.

| RRNCF             |   | Rotate Right f (No Carry) |                      |  |  |      |      |      |      |        |                   |              |                      |
|-------------------|---|---------------------------|----------------------|--|--|------|------|------|------|--------|-------------------|--------------|----------------------|
| Syntax:           | RRNCF f {,d {,a}}   |                           |                      |  |  |      |      |      |      |        |                   |              |                      |
| Operands:         | $0 \leq f \leq 255$<br>$d \in [0,1]$<br>$a \in [0,1]$   |                           |                      |  |  |      |      |      |      |        |                   |              |                      |
| Operation:        | $(f < n) \rightarrow \text{dest} < n - 1 >$ ,<br>$(f < 0) \rightarrow \text{dest} < 7 >$  |                           |                      |  |  |      |      |      |      |        |                   |              |                      |
| Status Affected:  | N, Z  |                           |                      |  |  |      |      |      |      |        |                   |              |                      |
| Encoding:         | <table><tr><td>0100</td><td>00da</td><td>ffff</td><td>ffff</td></tr></table>  |                           |                      |  |  | 0100 | 00da | ffff | ffff |        |                   |              |                      |
| 0100              | 00da  | ffff                      | ffff                 |  |  |      |      |      |      |        |                   |              |                      |
| Description:      | <p>The contents of register 'f' are rotated one bit to the right. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). If 'a' is '0', the Access Bank will be selected (default), overriding the BSR value. If 'a' is '1', then the bank will be selected as per the BSR value. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <b>Section 36.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</b> for details.</p> <div></div> |                           |                      |  |  |      |      |      |      |        |                   |              |                      |
| Words:            | 1   |                           |                      |  |  |      |      |      |      |        |                   |              |                      |
| Cycles:           | 1   |                           |                      |  |  |      |      |      |      |        |                   |              |                      |
| Q Cycle Activity: | <table><tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr><tr><td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr></table>  |                           |                      |  |  | Q1   | Q2   | Q3   | Q4   | Decode | Read register 'f' | Process Data | Write to destination |
| Q1                | Q2  | Q3                        | Q4                   |  |  |      |      |      |      |        |                   |              |                      |
| Decode            | Read register 'f'   | Process Data              | Write to destination |  |  |      |      |      |      |        |                   |              |                      |

**Example 1:** RRNCF REG, 1, 0

Before Instruction  
 REG = 1101 0111  
 After Instruction  
 REG = 1110 1011

**Example 2:** RRNCF REG, 0, 0

Before Instruction  
 W = ?  
 REG = 1101 0111  
 After Instruction  
 W = 1110 1011  
 REG = 1101 0111

| SETF              |  | Set f             |              |                    |  |      |      |      |      |
|-------------------|--|-------------------|--------------|--------------------|--|------|------|------|------|
| Syntax:           | SETF f {,a}  |                   |              |                    |  |      |      |      |      |
| Operands:         | $0 \leq f \leq 255$<br>$a \in [0,1]$   |                   |              |                    |  |      |      |      |      |
| Operation:        | FFh $\rightarrow$ f  |                   |              |                    |  |      |      |      |      |
| Status Affected:  | None   |                   |              |                    |  |      |      |      |      |
| Encoding:         | <table border="1"><tr><td>0110</td><td>100a</td><td>ffff</td><td>ffff</td></tr></table>  |                   |              |                    |  | 0110 | 100a | ffff | ffff |
| 0110              | 100a   | ffff              | ffff         |                    |  |      |      |      |      |
| Description:      | <p>The contents of the specified register are set to FFh.</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <b>Section 36.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</b> for details.</p> |                   |              |                    |  |      |      |      |      |
| Words:            | 1  |                   |              |                    |  |      |      |      |      |
| Cycles:           | 1  |                   |              |                    |  |      |      |      |      |
| Q Cycle Activity: |  |                   |              |                    |  |      |      |      |      |
|                   | Q1   | Q2                | Q3           | Q4                 |  |      |      |      |      |
|                   | Decode   | Read register 'f' | Process Data | Write register 'f' |  |      |      |      |      |

**Example:** SETF REG, 1

Before Instruction  
 REG = 5Ah  
 After Instruction  
 REG = FFh

## 37.6 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.

## 37.7 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.

## 37.8 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.

## 37.9 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™).

## 37.10 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.

---

**Note the following details of the code protection feature on Microchip devices:**

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

---

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

*Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELoq® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.*

**QUALITY MANAGEMENT SYSTEM**  
**CERTIFIED BY DNV**  
**== ISO/TS 16949 ==**

### Trademarks

The Microchip name and logo, the Microchip logo, AnyRate, AVR, AVR logo, AVR Freaks, BeaconThings, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, Helder, JukeBlox, KEELoq, KEELoq 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.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, BodyCom, CodeGuard, CryptoAuthentication, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming, ICSP, Inter-Chip Connectivity, JitterBlocker, KleerNet, KleerNet logo, Mindi, MiWi, motorBench, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PureSilicon, QMatrix, RightTouch logo, REAL ICE, Ripple Blocker, SAM-ICE, Serial Quad I/O, SMART-I.S., SQI, SuperSwitcher, SuperSwitcher II, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

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.

© 2017, Microchip Technology Incorporated, All Rights Reserved.

ISBN: 978-1-5224-1636-4