



Welcome to 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 "<u>Embedded -</u> <u>Microcontrollers</u>"

#### Details

| Product Status             | Active   |
|----------------------------|--|
| Core Processor             | PIC  |
| Core Size                  | 8-Bit  |
| Speed                      | 32MHz  |
| Connectivity               | I <sup>2</sup> C, LINbus, SPI, UART/USART                                  |
| Peripherals                | Brown-out Detect/Reset, POR, PWM, WDT                                      |
| Number of I/O              | 25   |
| Program Memory Size        | 7KB (4K x 14)  |
| Program Memory Type        | FLASH  |
| EEPROM Size                | 256 x 8  |
| RAM Size                   | 512 x 8  |
| Voltage - Supply (Vcc/Vdd) | 2.3V ~ 5.5V  |
| Data Converters            | A/D 24x10b; D/A 1x5b   |
| Oscillator Type            | Internal   |
| Operating Temperature      | -40°C ~ 85°C (TA)  |
| Mounting Type              | Surface Mount  |
| Package / Case             | 28-VQFN Exposed Pad  |
| Supplier Device Package    | 28-QFN (6x6)   |
| Purchase URL               | https://www.e-xfl.com/product-detail/microchip-technology/pic16f18854-i-ml |

Email: info@E-XFL.COM

Address: Room A, 16/F, Full Win Commercial Centre, 573 Nathan Road, Mongkok, Hong Kong

### **Table of Contents**

| 1.0  | Device Overview   |  |
|------|---|--|
| 2.0  | Enhanced Mid-Range CPU  |  |
| 3.0  | Memory Organization   |  |
| 4.0  | Device Configuration  |  |
| 5.0  | Resets  |  |
| 6.0  | Oscillator Module (with Fail-Safe Clock Monitor)                          |  |
| 7.0  | Interrupts  |  |
| 8.0  | Power-Saving Operation Modes  |  |
| 9.0  | Windowed Watchdog Timer (WWDT)  |  |
| 10.0 | Nonvolatile Memory (NVM) Control  |  |
| 11.0 | Cyclic Redundancy Check (CRC) Module                                      |  |
| 12.0 | I/O Ports   |  |
| 13.0 | Peripheral Pin Select (PPS) Module  |  |
| 14.0 | Peripheral Module Disable   |  |
| 15.0 | Interrupt-On-Change   |  |
| 16.0 | Fixed Voltage Reference (FVR)   |  |
| 17.0 | Temperature Indicator Module  |  |
| 18.0 | Comparator Module   |  |
| 19.0 | Pulse-Width Modulation (PWM)  |  |
| 20.0 | Complementary Waveform Generator (CWG) Module                             |  |
| 21.0 | Zero-Cross Detection (ZCD) Module   |  |
| 22.0 | Configurable Logic Cell (CLC)   |  |
| 23.0 | Analog-to-Digital Converter With Computation (ADC2) Module                |  |
| 24.0 | Numerically Controlled Oscillator (NCO) Module                            |  |
| 25.0 | 5-Bit Digital-to-Analog Converter (DAC1) Module                           |  |
| 26.0 | Data Signal Modulator (DSM) Module  |  |
| 27.0 | Timer0 Module   |  |
| 28.0 | Timer1/3/5 Module with Gate Control                                       |  |
| 29.0 | Timer2/4/6 Module   |  |
| 30.0 | Capture/Compare/PWM Modules   |  |
| 31.0 | Master Synchronous Serial Port (MSSP) Modules                             |  |
| 32.0 | Signal Measurement Timer (SMT)  |  |
| 33.0 | Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART) |  |
| 34.0 | Reference Clock Output Module   |  |
| 35.0 | In-Circuit Serial Programming™ (ICSP™)                                    |  |
| 36.0 | Instruction Set Summary   |  |
| 37.0 | Electrical Specifications   |  |
| 38.0 | DC and AC Characteristics Graphs and Charts                               |  |
| 39.0 | Development Support   |  |
| 40.0 | Packaging Information   |  |
| Data | Sheet Revision History  |  |

#### **TABLE 1-2:** PIC16F18854 PINOUT DESCRIPTION (CONTINUED)

| Name                              | Function  | Input<br>Type | Output Type       | Description                                  |
|-----------------------------------|-----------|---------------|-------------------|--|
| OUT <sup>(2)</sup>                | CWG3C     | —             | CMOS/OD           | Complementary Waveform Generator 3 output C. |
|                                   | CWG3D     | —             | CMOS/OD           | Complementary Waveform Generator 3 output D. |
|                                   | CLC1OUT   | —             | CMOS/OD           | Configurable Logic Cell 1 output.            |
|                                   | CLC2OUT   | —             | CMOS/OD           | Configurable Logic Cell 2 output.            |
|                                   | CLC3OUT   | —             | CMOS/OD           | Configurable Logic Cell 3 output.            |
|                                   | CLC4OUT   | —             | CMOS/OD           | Configurable Logic Cell 4 output.            |
|                                   | NCO1      | —             | CMOS/OD           | Numerically Controller Oscillator output.    |
|                                   | CLKR      | —             | CMOS/OD           | Clock Reference module output.               |
| Legend: AN = Analog input or outp | ut CMOS = |               | mnatihle innut or | outout OD = Open-Drain                       |

TTL = TTL compatible input ST = Schmitt Trigger input with CMOS levels

1<sup>2</sup>C = Schmitt Trigger input with I<sup>2</sup>C

HV = High Voltage XTAL = Crystal levels 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. Refer to Table 13-1 for details on which PORT pins may be used for this signal.

2: All output signals shown in this row are PPS remappable. These signals may be mapped to output onto one of several PORTx pin options as described in Table 13-3.

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 3: PPS output registers.

These pins are configured for  $l^2C$  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  $l^2C$  specific or SMBus input buffer thresholds. 4:

### 2.1 Automatic Interrupt Context Saving

During interrupts, certain registers are automatically saved in shadow registers and restored when returning from the interrupt. This saves stack space and user code. See **Section 7.5 "Automatic Context Saving"** for more information.

### 2.2 16-Level Stack with Overflow and Underflow

These devices have a hardware stack memory 15 bits wide and 16 words deep. A Stack Overflow or Underflow will set the appropriate bit (STKOVF or STKUNF) in the PCON register, and if enabled, will cause a software Reset. See **Section 3.4 "Stack"** for more details.

### 2.3 File Select Registers

There are two 16-bit File Select Registers (FSR). FSRs can access all file registers and program memory, which allows one Data Pointer for all memory. When an FSR points to program memory, there is one additional instruction cycle in instructions using INDF to allow the data to be fetched. General purpose memory can now also be addressed linearly, providing the ability to access contiguous data larger than 80 bytes. There are also new instructions to support the FSRs. See **Section 3.5 "Indirect Addressing"** for more details.

### 2.4 Instruction Set

There are 49 instructions for the enhanced mid-range CPU to support the features of the CPU. See **Section 36.0 "Instruction Set Summary"** for more details.

### 3.3 PCL and PCLATH

The Program Counter (PC) is 15 bits wide. The low byte comes from the PCL register, which is a readable and writable register. The high byte (PC<14:8>) is not directly readable or writable and comes from PCLATH. On any Reset, the PC is cleared. Figure 3-3 shows the five situations for the loading of the PC.

FIGURE 3-3: LOADING OF PC IN DIFFERENT SITUATIONS



#### 3.3.1 MODIFYING PCL

Executing any instruction with the PCL register as the destination simultaneously causes the Program Counter PC<14:8> bits (PCH) to be replaced by the contents of the PCLATH register. This allows the entire contents of the program counter to be changed by writing the desired upper seven bits to the PCLATH register. When the lower eight bits are written to the PCL register, all 15 bits of the program counter will change to the values contained in the PCLATH register.

### 3.3.2 COMPUTED GOTO

A computed GOTO is accomplished by adding an offset to the program counter (ADDWF PCL). When performing a table read using a computed GOTO method, care should be exercised if the table location crosses a PCL memory boundary (each 256-byte block). Refer to Application Note AN556, *"Implementing a Table Read"* (DS00556).

### 3.3.3 COMPUTED FUNCTION CALLS

A computed function CALL allows programs to maintain tables of functions and provide another way to execute state machines or look-up tables. When performing a table read using a computed function CALL, care should be exercised if the table location crosses a PCL memory boundary (each 256-byte block).

If using the CALL instruction, the PCH<2:0> and PCL registers are loaded with the operand of the CALL instruction. PCH<6:3> is loaded with PCLATH<6:3>.

The CALLW instruction enables computed calls by combining PCLATH and W to form the destination address. A computed CALLW is accomplished by loading the W register with the desired address and executing CALLW. The PCL register is loaded with the value of W and PCH is loaded with PCLATH.

### 3.3.4 BRANCHING

The branching instructions add an offset to the PC. This allows relocatable code and code that crosses page boundaries. There are two forms of branching, BRW and BRA. The PC will have incremented to fetch the next instruction in both cases. When using either branching instruction, a PCL memory boundary may be crossed.

If using BRW, load the W register with the desired unsigned address and execute BRW. The entire PC will be loaded with the address PC + 1 + W.

If using BRA, the entire PC will be loaded with PC + 1, the signed value of the operand of the BRA instruction.

### 3.4 Stack

All devices have a 16-level x 15-bit wide hardware stack (refer to Figure 3-4 through Figure 3-7). The stack space is not part of either program or data space. The PC is PUSHed onto the stack when CALL or CALLW instructions are executed or an interrupt causes a branch. The stack is POPed in the event of a RETURN, RETLW or a RETFIE instruction execution. PCLATH is not affected by a PUSH or POP operation.

The stack operates as a circular buffer if the STVREN bit is programmed to '0' (Configuration Words). This means that after the stack has been PUSHed sixteen times, the seventeenth PUSH overwrites the value that was stored from the first PUSH. The eighteenth PUSH overwrites the second PUSH (and so on). The STKOVF and STKUNF flag bits will be set on an Overflow/Underflow, regardless of whether the Reset is enabled.

Note 1: There are no instructions/mnemonics called PUSH or POP. These are actions that occur from the execution of the CALL, CALLW, RETURN, RETLW and RETFIE instructions or the vectoring to an interrupt address.

### 3.4.1 ACCESSING THE STACK

The stack is available through the TOSH, TOSL and STKPTR registers. STKPTR is the current value of the Stack Pointer. TOSH:TOSL register pair points to the TOP of the stack. Both registers are read/writable. TOS is split into TOSH and TOSL due to the 15-bit size of the PC. To access the stack, adjust the value of STKPTR, which will position TOSH:TOSL, then read/write to TOSH:TOSL. STKPTR is five bits to allow detection of overflow and underflow.

| Note: | Care should be taken when modifying the |
|-------|---|
|       | STKPTR while interrupts are enabled.    |

During normal program operation, CALL, CALLW and interrupts will increment STKPTR while RETLW, RETURN, and RETFIE will decrement STKPTR. At any time, STKPTR can be inspected to see how much stack is left. The STKPTR always points at the currently used place on the stack. Therefore, a CALL or CALLW will increment the STKPTR and then write the PC, and a return will unload the PC and then decrement the STKPTR.

Reference Figure 3-4 through Figure 3-7 for examples of accessing the stack.



# 5.13 Register Definitions: Power Control

### REGISTER 5-2: PCON0: POWER CONTROL REGISTER 0

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

| ared by hardware  | e  | HS = Bit is set by hardware  |  |  |  |  |  |  |
|---|--|--|--|--|--|--|--|--|
| bit \   | N = Writable bit   | U = Unimplemented bit, read as '0'   |  |  |  |  |  |  |
| anged >   | <pre>c = Bit is unknown</pre>  | -m/n = Value at POR and BOR/Value at all other Resets  |  |  |  |  |  |  |
| 4   | 0' = Bit is cleared  | q = Value depends on condition   |  |  |  |  |  |  |
|   |  |  |  |  |  |  |  |  |
| <b>STKOVF:</b> Stack $1 = A$ Stack $O_{1}$  | k Overflow Flag bit<br>/erflow occurred  | cleared by firmware  |  |  |  |  |  |  |
|   | k Indorflow Elog bit   |  |  |  |  |  |  |  |
| 1 = A Stack Ur  | derflow occurred   |  |  |  |  |  |  |  |
| 0 = A Stack Ur  | nderflow has not occurred o  | r cleared by firmware  |  |  |  |  |  |  |
| WDTWV: WDT  | Window Violation Flag bit  |  |  |  |  |  |  |  |
| 1 = A WDT Window Violation Reset has not occurred or set by firmware  |  |  |  |  |  |  |  |  |
| 0 = A WD1 Wir   | Vindow Violation Reset has occurred (a CLRWDT instruction was executed either without per window or outside the window (cleared by bardware)   |  |  |  |  |  |  |  |
| <b>RWDT</b> : Watchd  | log Timer Reset Flag hit   |  |  |  |  |  |  |  |
| 1 = A Watchdog  | g Timer Reset has not occu   | rred or set to '1' by firmware   |  |  |  |  |  |  |
| 0 = A Watchdog  | g Timer Reset has occurred   | I (cleared by hardware)  |  |  |  |  |  |  |
| RMCLR: MCLF   | R Reset Flag bit   |  |  |  |  |  |  |  |
| 1 = A MCLR Reset has not occurred or set to '1' by firmware<br>A = A MCLR Reset has accurred (cleared by hardware)          |  |  |  |  |  |  |  |  |
|   | eset has occurred (cleared)  | by hardware)   |  |  |  |  |  |  |
| <b>RI:</b> RESET Instruction Flag bit<br>$1 = \mathbf{A}$ RESET instruction has not been executed or set to '1' by firmware |  |  |  |  |  |  |  |  |
| 0 = A  RESET instruction has been executed (cleared by hardware)  |  |  |  |  |  |  |  |  |
| POR: Power-or   | n Reset Status bit   |  |  |  |  |  |  |  |
| 1 = No Power-on Reset occurred  |  |  |  |  |  |  |  |  |
| 0 = A Power-or  | n Reset occurred (must be s  | set in software after a Power-on Reset occurs)   |  |  |  |  |  |  |
| BOR: Brown-ou   | ut Reset Status bit  |  |  |  |  |  |  |  |
| $\perp$ = No Brown-0  | out Reset occurred (must be  | set in software after a Power-on Reset or Brown-out Reset  |  |  |  |  |  |  |
| occurs)   |  |  |  |  |  |  |  |  |
|   | STKOVF: Stack<br>anged<br>STKOVF: Stack<br>1 = A Stack Ov<br>0 = A Stack Ov<br>STKUNF: Stack<br>1 = A Stack Ur<br>0 = A Stack Ur<br>WDTWV: WDT<br>1 = A WDT Wir<br>0 = A WDT Wir<br>1 = A WDT Wir<br>0 = A WDT Wir<br>0 = A WDT Wir<br>1 = A WDT Wir<br>0 = A WDT Wir<br>1 = A WOT WIR<br>1 = A RESET INSTR<br>1 = A RESET INSTR<br>1 = NO POWER-OR<br>0 = A POWER-OR<br>1 = NO BROWN-OR<br>0 = A BRO | ared by hardware         bit       W = Writable bit         anged       x = Bit is unknown         '0' = Bit is cleared         STKOVF: Stack Overflow Flag bit         1 = A Stack Overflow occurred         0 = A Stack Overflow has not occurred or         STKUNF: Stack Underflow has not occurred or         STKUNF: Stack Underflow occurred         0 = A Stack Underflow occurred         0 = A Stack Underflow occurred         0 = A Stack Underflow has not occurred or         WDTWF: WDT Window Violation Flag bit         1 = A WDT Window Violation Reset has not         0 = A WDT Window Violation Reset has or         arming the window or outside the window         RWDT: Watchdog Timer Reset Flag bit         1 = A Watchdog Timer Reset Flag bit         1 = A MCLR Reset has not occurred         0 = A MCLR Reset has not occurred or set         0 = A MCLR Reset has not occurred or set         0 = A RESET Instruction Flag bit         1 = No Power-on Reset Status bit         1 = No Power-on Reset occurred         0 = A Power-on Reset occurred         0 = A Power-on Reset occurred         0 = A Brown-out Reset occurred         0 = A Brown-out Reset occurred (must be so occurs) |  |  |  |  |  |  |

|  | TABLE 5-5: | SUMMARY OF REGISTERS ASSOCIATED WITH RES | SETS |
|--|------------|--|------|
|--|------------|--|------|

| Name    | Bit 7  | Bit 6  | Bit 5           | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0  | Register<br>on Page |
|---------|--------|--------|-----------------|-------|-------|-------|-------|--------|---------------------|
| BORCON  | SBOREN |        |                 |       |       |       |       | BORRDY | 85                  |
| PCON0   | STKOVF | STKUNF | WDTWV           | RWDT  | RMCLR | RI    | POR   | BOR    | 90                  |
| STATUS  | _      | _      | _               | TO    | PD    | Z     | DC    | С      | 26                  |
| WDTCON0 | _      | _      | WDTPS<4:0> SWDT |       |       |       |       |        | 146                 |

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

# REGISTER 8-2: CPUDOZE: DOZE AND IDLE REGISTER

| R/W-0/u  | R/W/HC/HS-0/0  | R/W-0/0  | R/W-0/0                     | U-0              | R/W-0/0           | R/W-0/0            | R/W-0/0         |
|--|--|--|-----------------------------|------------------|-------------------|--------------------|-----------------|
| IDLEN  | DOZEN <sup>(1,2)</sup>   | ROI  | DOE                         | _                |                   | DOZE<2:0>          |                 |
| bit 7  |  |  | •                           |                  |                   |                    | bit 0           |
|  |  |  |                             |                  |                   |                    |                 |
| Legend:  |  |  |                             |                  |                   |                    |                 |
| R = Readable bi  | t  | W = Writable bit                                       | t                           | U = Unimplen     | nented bit, read  | as '0'             |                 |
| u = Bit is unchar  | nged   | x = Bit is unknow                                      | wn                          | -n/n = Value a   | t POR and BOI     | R/Value at all oth | er Resets       |
| '1' = Bit is set   |  | '0' = Bit is cleare                                    | ed                          |                  |                   |                    |                 |
| bit 7 <b>IDLEN:</b> Idle Enable bit<br>1 = A SLEEP instruction inhibits the CPU clock, but not the peripheral clock(s)<br>0 = A SLEEP instruction places the device into full Sleep mode |  |  |                             |                  |                   |                    |                 |
| bit 6  | <ul> <li>DOZEN: Doze Enable bit<sup>(1,2)</sup></li> <li>1 = The CPU executes instruction cycles according to DOZE setting</li> <li>0 = The CPU executes all instruction cycles (fastest, highest power operation)</li> </ul>  |  |                             |                  |                   |                    |                 |
| bit 5  | <b>ROI:</b> Recover-on-<br>1 = Entering the<br>0 = Interrupt entr  | Interrupt bit<br>Interrupt Service<br>y does not chang | Routine (ISR) n<br>ge DOZEN | nakes DOZEN      | = 0 bit, bringing | the CPU to full-s  | peed operation. |
| bit 4  | <b>DOE:</b> Doze on Exit bit<br>1 = Executing RETFIE makes DOZEN = 1, bringing the CPU to reduced speed operation.<br>0 = RETFIE does not change DOZEN   |  |                             |                  |                   |                    |                 |
| bit 3  | Unimplemented:   | Read as '0'  |                             |                  |                   |                    |                 |
| bit 2-0  | DOZE<2:0>: Rational content of the second se | o of CPU Instruc                                       | tion Cycles to P            | eripheral Instru | ction Cycles      |                    |                 |

- **Note** 1: When ROI = 1 or DOE = 1, DOZEN is changed by hardware interrupt entry and/or exit.
  - 2: Entering ICD overrides DOZEN, returning the CPU to full execution speed; this bit is not affected.

| Name    | Bit 7   | Bit 6       | Bit 5      | Bit 4  | Bit 3     | Bit 2     | Bit 1     | Bit 0   | Register<br>on Page |  |
|---------|---------|-------------|------------|--------|-----------|-----------|-----------|---------|---------------------|--|
| OSCCON1 | —       | 1           | NOSC<2:0>  |        |           | NDIV<3:0> |           |         |                     |  |
| OSCCON2 | —       | (           | COSC<2:0>  |        |           | CDIV<     | 3:0>      |         | 102                 |  |
| OSCCON3 | CSWHOLD | SOSCPWR     | _          | ORDY   | NOSCR     | —         | —         | _       | 103                 |  |
| PCON0   | STKOVF  | STKUNF      | WDTWV      | RWDT   | RMCLR     | RI        | POR       | BOR     | 90                  |  |
| STATUS  | —       | —           | _          | TO     | PD        | Z         | DC        | С       | 26                  |  |
| WDTCON0 | —       | -           |            |        | WDTPS<4:0 | )>        |           | SEN     | 146                 |  |
| WDTCON1 | —       | v           | VDTCS<2:0> |        | —         | WI        | NDOW<2:0> | >       | 146                 |  |
| WDTPSL  |         | PSCNT<7:0>  |            |        |           |           |           |         |                     |  |
| WDTPSH  |         | PSCNT<15:8> |            |        |           |           |           |         | 146                 |  |
| WDTTMR  |         |             | WDTTM      | R<4:0> |           | STATE     | PSCNT     | <17:16> | 146                 |  |

### TABLE 9-3: SUMMARY OF REGISTERS ASSOCIATED WITH WATCHDOG TIMER

**Legend:** – = unimplemented locations read as '0'. Shaded cells are not used by Watchdog Timer.

### TABLE 9-4: SUMMARY OF CONFIGURATION WORD WITH WATCHDOG TIMER

| Name    | Bits | Bit -/7 | Bit -/6 | Bit 13/5   | Bit 12/4 | Bit 11/3 | Bit 10/2 | Bit 9/1    | Bit 8/0  | Register<br>on Page |
|---------|------|---------|---------|------------|----------|----------|----------|------------|----------|---------------------|
|         | 13:8 | _       |         | FCMEN      |          | CSWEN    | SWEN — — |            | CLKOUTEN | 74                  |
| CONFIGT | 7:0  | _       | F       | RSTOSC<2:0 | >        | _        | F        | EXTOSC<2:0 | >        | 74                  |

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

#### EXAMPLE 10-5: DEVICE ID ACCESS

; This write routine assumes the following:

| BANKSEL NVMADRH<br>MOVF ADDRH, M<br>MOVF ADDRL, M<br>MOVF ADDRL, M<br>MOVF ADDRL, M<br>MOVWF NVMADRH<br>MOVWF NVMADRH<br>MOVUW LOW DATA_ADDR ; Load initial data address<br>MOVWF FSR0L<br>MOVWF FSR0L<br>BCF NVMCONI, WRERSS ; Set PFM as write location<br>BSF NVMCONI, WREN ; Enable writes<br>BSF NVMCONI, WREN ; Load first data byte<br>MOVIF FSR0H<br>MOVF FSR0H<br>NOVF SR0H+<br>MOVF NVMADRL, M<br>XORLW 0xIF ; Check if lower bits of address are 00000<br>ANDLW 0xIF ; Check if lower bits of address are 00000<br>ANDLW 0xIF ; Load first data byte<br>MOVF NVMADRL, M<br>XORLW 0xIF ; load sof 032 addresses<br>BTFSC STATUS, Z ; Last of 32 words?<br>GOTO STATI_WRITE ; If so, go write latches into memory<br>CALL UNLOCK_SEQ ; If not, go load latch<br>INCF NVMADRL, F ; Increment address<br>GOTO LOOP<br>START_WRITE<br>BCF NVMCONI, LNLO ; Latch writes complete, now write memory<br>CALL UNLOCK_SEQ ; Perform required unlock sequence<br>BCF NVMCONI, MREN ; Disable writes<br>UNLOCK_SEQ<br>MOVLW SSh<br>BCF INTCON, GIE ; Disable interrupts<br>MOVLW AAA<br>BCF INTCON, GIE ; Disable interrupts<br>MOVLW AAA<br>BCF NVMCONI, MREN ; Begin unlock sequence<br>MOVLW AAA<br>BCF NVMCONI, MR<br>BCF N  | ; 1. 6<br>; 2. E<br>; store<br>; 3. A<br>; 4. A<br>; 5. N | 4 bytes of data are lo<br>ach word of data to be<br>d in little endian for<br>valid starting addres<br>DDRH and ADDRL are loo<br>VM interrupts are not | baded, starting at the a<br>e written is made up of<br>cmat<br>as (the least significan<br>cated in common RAM (loc<br>taken into account | ddress in DATA_ADDR<br>two adjacent bytes in DATA_ADDR,<br>t bits = 00000) is loaded in ADDRH:ADDRL<br>ations 0x70 - 0x7F) |
|---|---|--|---|--|
| NOVFADDEL, WMOVFADDEL, WMOVFADDEL, WMOVFNVMADRHMOVFNVMADRHMOVWFNVMADRHMOVWFFSR01MOVWFFSR01MOVWFFSR01MOVWFFSR01BSFNVMCON1, NVMRENSBSFNVMCON1, NVRENSBSFNVMCON1, NVRENSBSFNVMCON1, NVRENSLOOPFSR01MOVIWFSR01MOVIWFSR01MOVIWSSNSTART_WRITEIf so, go write latches into memoryCALLNUMCON1, NRENBCFNUMCON1, NRENSCALLNUMCON1, NRENMOVIWSSNMOVIWSSNMOVIWSSNMOVIWSSNMOVIWNUMCON1, NRENSCALLNUMCON1, NRENBCFNUMCON1, NRENSCALLNUMCON1, NRENBCFNUMCON1, NRENSSNNUMCON1, NRENBCF  |   | BANKSEL  | NVMADRH   |  |
| <pre>MOWF NVMADEH ; Load initial address<br/>MOVF ADDRL,W<br/>MOVF NVMADEL<br/>MOVWF VIEW<br/>MOVWF FSROL<br/>MOVWF FSROL<br/>BCF NVMCONI,WIERDS ; Set PFM as write location<br/>BSF NVMCONI,WRENS ; Set PFM as write location<br/>BSF NVMCONI,WREN ; Enable writes<br/>BSF NVMCONI,WREN ; Enable write latches<br/>LOOP<br/>NOVIW FSROH<br/>MOVWF YSROH<br/>MOVWF NVMADATL ; Load first data byte<br/>MOVWF NVMADATL ; Load first data byte<br/>MOVWF NVMADATL ; Load second data byte<br/>MOVWF NVMADATH ; Load second data byte<br/>MOVF NVMADATH ; Load second data byte<br/>NOVF NVMADAL,W<br/>XOELW 0x1F ; Load of 32 words?<br/>GOTO START_WRITE ; If so, go write latches into memory<br/>CALL UNLOCK_SEQ ; If not, go load latch<br/>INCP NVMADAL,F ; Increment address<br/>GOTO LOOP<br/>START_WRITE<br/>BCF NVMCONI,WREN ; Disable writes<br/>UNLOCK_SEQ<br/>UNLOCK_SEQ<br/>NVMCW S5h<br/>BCF UNNCCNI,WREN ; Disable interrupts<br/>MOVWF NVMCONI,WREN ; Disable interrupts<br/>MOVWF NVMCONI,WREN ; Begin unlock sequence<br/>MOVWW AAh<br/>MOVWF NVMCONI,WR<br/>BSF UNVMCONI,WR<br/>BSF UNVMCONI,WR<br/>B</pre>  |   | MOVF   | ADDRH,W   |  |
| <pre>NOVF ADDEL,W<br/>NOVWF NVMADRL<br/>NOVIW LOW DATA_ADDR ; Load initial data address<br/>NOVWF FSROL<br/>NOVWF FSROL<br/>BCF NVMCONI,NVMREGS ; Set PFM as write location<br/>BSF NVMCONI,NRENS ; Set PFM as write location<br/>BSF NVMCONI,NRENS ; Load only write latches<br/>DOP<br/>NOVIW FSROH<br/>NOVWF NVMCONI,NERN ; Load first data byte<br/>NOVWF NVMCONI, LWLO ; Load first data byte<br/>NOVWF NVMDATL ; Load first data byte<br/>NOVWF NVMDATL ; Load second data byte<br/>NOVF NVMDATH ; Load second data byte<br/>NOVF NVMDATH ; Load second data byte<br/>NOVF NVMDATH ; Load second data byte<br/>STRET_NOVF NVMADRL,W<br/>XORLW 0x1F ; Check if lower bits of address are 00000<br/>ANDLW 0x1F ; If so, go write latches into memory<br/>GOTO START_WRITE ; If so, go write latches into memory<br/>CALL UNLOCK_SEQ ; If not, go load latch<br/>INCF NVMADRL,F ; Increment address<br/>GOTO LOOP<br/>START_WRITE<br/>BCF NVMCONI,WREN ; Latch writes complete, now write memory<br/>CALL UNLOCK_SEQ ; Perform required unlock sequence<br/>BCF NVMCONI,WREN ; Disable writes<br/>UNLOCK_SEQ<br/>NULOCK_SEQ<br/>NULOCK_SEQ<br/>NULOCK_SEQ<br/>NULOCK_SEQ<br/>NULOCK_SEQ<br/>NOVWF NVMCONI,WREN ; Disable interrupts<br/>NOVWF NVMCONI,WREN ; Disable interrupts<br/>NOVWF NVMCONI,WR<br/>BSF NVMCONI,WR<br/>BSF</pre> |   | MOVWF  | NVMADRH   | ; Load initial address   |
| MOVNENVMADRLMOVLMLOW DATA_ADDR; Load initial data addressMOVNFFSR0LMOVNFFSR0HBCFNVMCON1,NVMREGS; Set PFM as write locationBSFNVMCON1,NVMREGS; Set PFM as write locationBSFNVMCON1,NVMREGS; Set PFM as write locationBSFNVMCON1,NVMREGS; Load only write latchesLOOPMOVINFSR0++MOVFNVMDATL; Load first data byteMOVFNVMDATH; Load second data byteMOVFNVMDATH; Load if lower bits of address are 00000ANDLWOxIF; and if on last of 32 addressesBTFSCSTATUS,Z; Last of 32 words?GOTOSTATUS,Z; Inorement addressCALLUNLOCK_SEQ; If not, go load latchINCFNVMCON1,LWLO; Latch writes complete, now write memoryCALLUNLOCK_SEQ; Perform required unlock sequenceGOTOLOOPSTART_WRITEUNLOCK_SEQ; Perform required unlock sequenceBCFNVMCON1,WREN; Disable writesUNLOCK_SEQS5hBCFNVMCON2BCFNVMCON2; Begin unlock sequenceMOVINAhMOVINPNVMCON1,WRENBSFNVMCON1,WRENBSFNVMCON1,WRBSFNVMCON1,WRBSFNVMCON1,WRBSFNVMCON2BSFNVMCON2BSFNVMCON1,WRBSFNVMCON1,WRBSFN  |   | MOVF   | ADDRL,W   |  |
| <pre>MOULW LOW DATA_ADDR ; Load initial data address<br/>MOUWF FSR0L<br/>MOULW HIGH DATA_ADDR<br/>MOUWF FSR0H<br/>BCF NVMCONI,NVMERGS ; Set PFM as write location<br/>BSF NVMCONI,NVMERGS ; Set PFM as write location<br/>BSF NVMCONI,NVMERGS ; Load only write latches<br/>BSF NVMCONI,NVMERGS ; Load only write latches<br/>NOVIW FSR0++<br/>MOUVF NVMDATL ; Load first data byte<br/>NOVF NVMADRL, W<br/>NOVF NVMADRL, W<br/>XORLW 0x1F ; Check if lower bits of address are 00000<br/>ANDLW 0x1F ; and if on last of 32 addresses<br/>BTFSC STATUS, Z ; Last of 32 words?<br/>GOTO START_WRITE ; If so, go Write latches into memory<br/>CALL UNLOCK_SEQ ; If not, go load latch<br/>INCF NVMADRL,F ; Increment address<br/>GOTO START_WRITE ; If so, go Write latches into memory<br/>CALL UNLOCK_SEQ ; If not, go load latch<br/>INCF NVMADRL,F ; Increment address<br/>BCF NVMCONI,LMLO ; Latch writes complete, now write memory<br/>CALL UNLOCK_SEQ ; Perform required unlock sequence<br/>BCF NVMCONI,LMLO ; Latch writes complete, now write memory<br/>CALL NOCK_SEQ ; Perform required unlock sequence<br/>BCF NVMCONI,LMLO ; Latch writes complete, now write memory<br/>CALL NUNCCK_SEQ ; Perform required unlock sequence<br/>BCF NVMCONI, MREN ; Disable interrupts<br/>MOVUW Sh<br/>BCF NVMCONI, MREN ; Disable interrupts<br/>MOVUWF NVMCON2 ; Begin unlock sequence complete, re-enable interrupts<br/>MOVUW AAh<br/>MOVWF NVMCON2<br/>BSF NVMCONI,WR SSF NVMCONI SSF NVMCONI,WR SSF NVMCONI,WR SSF NVMCONI,W</pre>  |   | MOVWF  | NVMADRL   |  |
| MOVWF FSR0L<br>MOVWF FIGH DATA_ADDR<br>MOVWF FSR0H<br>BCF NVMCON1,NVRREGS ; Set PFM as write location<br>BSF NVMCON1,NVREGS ; Set PFM as write location<br>MOVUF NVMCON1,NVREGS ; Load only write latches<br>NOVWF NVMCON1,NVREGS ; Load only write latches<br>NOVWF NVMCON1,NVREGS ; Load first data byte<br>MOVWF NVMDATL ; Load first data byte<br>MOVWF NVMDATL ; Load second data byte<br>MOVVF NVMADRL,W<br>XORLW 0x1F ; Check if lower bits of address are 00000<br>ANDLW 0x1F ; and if on last of 32 addresses<br>BTFSC STATUS,Z ; Last of 32 words?<br>GOTO START_WRITE ; If so, go write latches into memory<br>CALL ULLOCK_SEQ ; If not, go load latch<br>INCOF NVMADRL,F ; Increment address<br>GOTO LOOP<br>START_WRITE<br>BCF NVMCON1,LWLO ; Latch writes complete, now write memory<br>CALL UNLOCK_SEQ ; Perform required unlock sequence<br>BCF NVMCON1,WREN ; Disable writes<br>UNLOCK_SEQ<br>UNLOCK_SEQ<br>UNLOCK_SEQ<br>NVMCON1,WREN ; Disable interrupts<br>MOVUW AAA<br>MOVUWF NVMCON2 ; Begin unlock sequence<br>MOVUWF NVMCON1,WR<br>BSF NVMCON1,WR<br>BSF NVMCON1,WR<br>BSF NVMCON1,WR<br>BSF NVMCON1,WR  |   | MOVLW  | LOW DATA_ADDR   | ; Load initial data address  |
| MOULW HIGH DATA_ADDR<br>MOVWF FSR0H<br>BCF NVMCONI,NVMREGS : Set PFM as write location<br>BSF NVMCONI,WREN : Enable writes<br>BSF NVMCONI,LWLO : Load only write latches<br>LOOP<br>NOVIW FSR0++<br>MOVWF NVMDATL : Load first data byte<br>MOVWF NVMDATL : Load second data byte<br>MOVF NVMADRL,W<br>XORLW 0x1F : And if on last of 32 addresses<br>BTFSC STATUS,Z : Last of 32 words?<br>GOTO START_WRITE : If so, go write latches into memory<br>CALL UNLOCK_SEQ : If not, go load latch<br>INCF NVMCONI,LWLO : Latch writes complete, now write memory<br>CALL UNLOCK_SEQ : Disable interrupts<br>MOVUW 55h<br>BCF NVMCONI,WREN : Disable interrupts<br>MOVUW 55h<br>BCF INTCON,GIE : Disable interrupts<br>MOVUW ADA<br>MOVUW ADA<br>SFF INTCON,GIE : Disable interrupts<br>MOVUW ADA<br>MOVUW ADA<br>BSF NVMCONI,WR NVMCONI,WR<br>BSF NVMCONI,WR NVMCONI,WR<br>BSF NVMCONI,WR NVMCONI,WR<br>BSF NVMCONI,WR NVMCONI,WR<br>BSF NVMCONI,WR NC : Unlock sequence complete, re-enable interrupts<br>return   |   | MOVWF  | FSROL   |  |
| MOUWFFSRUHBCFNVMCON1, NVMRESS; Set PFM as write locationBSFNVMCON1, NUMESS; Load only write latchesLOOPMOVIWFSR0++MOVWFNVMDATL; Load first data byteMOVWFNVMDATL; Load second data byteMOVWFNVMDATH; Load second data byteMOVFNVMADRL,WXORLW0x1F; Check if lower bits of address are 00000ANDLW0x1F; Last of 32 addressesBTFSCSTATUS,Z; Last of 32 words?GOTOSTART_WRITE; If so, go write latches into memoryCALLUNLOCK_SEQ; If not, go load latchINCFNVMCON1,LWLO; Latch writes complete, now write memoryCALLUNLOCK_SEQ; Perform required unlock sequenceBCFNVMCON1,WREN; Disable interruptsMOVUWS5hBCFINTCON,GIE; Disable interruptsMOVUWSShBCFNVMCON1,WREN; Disable interruptsMOVUWSShBCFNVMCON1,WRMOVUWNVMCON1,WRBSFNVMCON1,WRBSFNVMCON1,WRBSFNVMCON1,WRBSFNVMCON1,WRBSFNVMCON1,WRBSFNVMCON1,WRBSFNVMCON1,WRBSFNVMCON1,WRBSFNVMCON1,WRBSFNVMCON1,WRBSFNVMCON1,WRBSFNVMCON1,WRBSFNVMCON1,WR<  |   | MOVLW  | HIGH DATA_ADDR  |  |
| BCF       NVMCONI,WMREGS       : Set PFM as write location         BSF       NVMCONI,WREN       : Enable writes         BSF       NVMCONI,LWLO       : Load only write latches         LOOP       MOVIW       FSR0++         MOVWF       NVMCATL       : Load first data byte         MOVWF       NVMADATL       : Load second data byte         MOVF       NVMADATH       : Load second data byte         MOVF       NVMADAL,W         XORLW       0x1F       : Check if lower bits of address are 00000         ANDLW       0x1F       : and if on last of 32 addresses         BTFSC       STATUS,Z       : Last of 32 words?         GOTO       START_WRITE       : If so, go write latches into memory         CALL       UNLOCK_SEQ       : If not, go load latch         INCP       NVMCON1,LWLO       : Latch writes complete, now write memory         CALL       UNLOCK_SEQ       : Perform required unlock sequence         BCF       NVMCON1,WREN       : Disable writes         UNLOCK_SEQ       : Perform required unlock sequence         MOVUW       S5h       BCF         BCF       INTCON,GIE       : Disable interrupts         MOVWF       NVMCON1,WR       BSF         BSF </th <th></th> <th>MOVWF</th> <th>FSROH</th> <th></th>  |   | MOVWF  | FSROH   |  |
| BSF NVMCONI,WREN : Enable writes<br>BSF NVMCONI,LWLO : Load only write latches<br>LOOP<br>MOVIW FSR0++<br>MOVWF NVMDATL : Load first data byte<br>MOVWF NVMDATH : Load second data byte<br>MOVF NVMADRL,W<br>XORLW 0x1F : and if on last of 32 addresses<br>BTFSC STATUS,Z : Last of 32 words?<br>GOTO START_WRITE : If so, go write latches into memory<br>CALL UNLOCK_SEQ : If not, go load latch<br>INCF NVMADRL,F : Increment address<br>GOTO LOOP<br>START_WRITE<br>BCF NVMCON1,LWLO : Latch writes complete, now write memory<br>CALL UNLOCK_SEQ : Perform required unlock sequence<br>BCF NVMCON1,WREN : Disable writes<br>UNLOCK_SEQ<br>MOVLW 55h<br>BCF INTCON,GIE : Disable interrupts<br>MOVWF NVMCON2 : Begin unlock sequence<br>MOVLW AAh<br>MOVWF NVMCON2<br>BSF NVMCON1,WR<br>BSF NVMCON1,WR<br>BSF NVMCON1,WR<br>BSF NVMCON1,WR<br>BSF NVMCON1,WR   |   | BCF  | NVMCON1,NVMREGS   | ; Set PFM as write location  |
| BSF NVMCON1,LWLO ; Load only write latches LOOP MOVIW FSR0++ MOVWF NVMDATL ; Load first data byte MOVIW FSR0++ MOVWF NVMDATH ; Load second data byte MOVF NVMDATH ; Load second data byte MOVF NVMDATH ; Load second data byte MOVF NVMADRL,W XORLW 0x1F ; Check if lower bits of address are 00000 ANDLW 0x1F ; and if on last of 32 addresses BTFSC STATUS,Z ; Last of 32 words? GOTO START_WRITE ; if so, go write latches into memory CALL UNLOCK_SEQ ; If not, go load latch INCF NVMADRL,F ; Increment address GOTO LOOP START_WRITE BCF NVMCON1,LWLO ; Latch writes complete, now write memory CALL UNLOCK_SEQ ; Perform required unlock sequence BCF NVMCON1,WREN ; Disable writes UNLOCK_SEQ MOVLW 55h BCF INTCON,GIE ; Disable interrupts MOVWF NVMCON2 BSF NVMCON1,WR BSF  |   | BSF  | NVMCON1,WREN  | ; Enable writes  |
| LOOP           LOOP         MOVIW         FSR0++           MOVWF         NVMDATL         ; Load first data byte           MOVWF         NVMDATH         ; Load second data byte           MOVWF         NVMADRL,W           XORLW         Ox1F         ; Check if lower bits of address are 00000           ANDLW         Ox1F         ; and if on last of 32 addresses           BTFSC         STATUS,Z         ; Last of 32 words?           GOTO         START_WRITE         ; If so, go write latches into memory           CALL         UNLOCK_SEQ         ; If not, go load latch           INCF         NVMADRL,F         ; Increment address           GOTO         LOOP         ; Disable writes           UNLOCK_SEQ         ; Disable writes           UNLOCK_SEQ         ; Disable interrupts           MOVUM         Sh           RCF         INTCON,GIE         ; Disable interrupts           MOVWF         NVMCON2         ; Begin unlock sequence           MOVWF         NVMCON1, NR         ; Begin unlock sequence           MOVWF         NVMCON1, NR         ; Begin unlock sequence           MOVWF         NVMCON1, NR         ; Begin unlock sequence           BSF         NVMCON1, NR         ; Sh           <   |   | BSF  | NVMCON1,LWLO  | ; Load only write latches  |
| MOVIW FSR0++<br>MOVWF NVMDATL ; Load first data byte<br>MOVWF NVMDATL ; Load second data byte<br>MOVWF NVMDATH ; Load second data byte<br>MOVF NVMADRL,W<br>XORLW 0x1F ; Check if lower bits of address are 00000<br>ANDLW 0x1F ; and if on last of 32 addresses<br>BTFSC STATUS,Z ; Last of 32 words?<br>GOTO START_WRITE ; If so, go write latches into memory<br>CALL UNLOCK_SEQ ; If not, go load latch<br>INCF NVMADRL,F ; Increment address<br>GOTO LOOP<br>START_WRITE<br>BCF NVMCON1,LWLO ; Latch writes complete, now write memory<br>CALL UNLOCK_SEQ ; Perform required unlock sequence<br>BCF NVMCON1,WREN ; Disable writes<br>UNLOCK_SEQ<br>MOVLW 55h<br>BCF INTCON,GIE ; Disable interrupts<br>MOVLW AAh<br>MOVWF NVMCON2<br>BSF NVMCON1,WR SEG ; Unlock sequence complete, re-enable interrupts<br>REF INTCON,GIE ; Unlock sequence complete, re-enable interrupts<br>NVMCON1,WR SEG ; Unlock sequence complete, re-enable interrupts<br>NOVWF NVMCON1,WR SEG ; Unlock sequence complete, re-enable interrupts<br>NOVWF SEG ; NVMCON1,WR SEG ; Unlock sequence complete, re-enable interrupts<br>NOVWF ; NVMCON1,WR SEG ; Unlock sequence complete, re-enable interrupts<br>NOVWF ; NVMCON1,WR SEG ; Unlock sequence complete, re-enable interrupts<br>NOVWF ; NVMCON1,WR SEG ; Unlock sequence complete, re-enable interrupts<br>NOVWF ; NVMCON1,WR SEG ; Unlock sequence complete, re-enable interrupts<br>NOVWF ; NVMCON1,WR SEG ; Unlock sequence complete, re-enable interrupts<br>NOV ; SEG ; SEG ; Unlock sequence complete, re-enable interrupts<br>NOV ; SEG ; SEG ; UNLOCK ; SEG ; UNLOC  | LOOP  |  |   |  |
| MOVWFNVMDATL: Load first data byteMOVIWFSR0++MOVWFNVMDATH; Load second data byteMOVFNVMADRL,WXORLW0x1F; Check if lower bits of address are 00000ANDLW0x1F; and if on last of 32 addressesBTFSCSTATUS,Z; Last of 32 words?GOTOSTART_WRITE; If so, go write latches into memoryCALLUNLOCK_SEQ; If not, go load latchINCFNVMADRL,F; Increment addressGOTOLOOPSTART_WRITESTART_WRITEECFNVMCON1,LWLOCALLUNLOCK_SEQ; Perform required unlock sequenceBCFNVMCON1,WREN; Disable writesUNLOCK_SEQINTCON,GIE; Disable interruptsMOVUWAhMOVWFMOVWFNVMCON1BEGIN unlock sequenceMOVWFNVMCON1,WRBEGIN unlock sequenceMOVWFNVMCON1; Disable interruptsMOVWFNVMCON1,WRBEGIN unlock sequenceMOVWFNVMCON1; Begin unlock sequenceMOVWFNVMCON1; Introve sequenceMOVWFNVMCON1,WR; Begin unlock sequenceBSFNVMCON1,WR; Shock sequence complete, re-enable interruptsREFINT   |   | MOVIW  | FSR0++  |  |
| MOVIW FSR0++<br>MOVWF NVMDATH ; Load second data byte<br>MOVF NVMADRL,W<br>XORLW 0x1F ; Check if lower bits of address are 00000<br>ANDLW 0x1F ; and if on last of 32 addresses<br>BTFSC STATUS,Z ; Last of 32 words?<br>GOTO START_WRITE ; If so, go write latches into memory<br>CALL UNLOCK_SEQ ; If not, go load latch<br>INCF NVMADRL,F ; Increment address<br>GOTO LOG<br>START_WRITE<br>BCF NVMCON1,LWLO ; Latch writes complete, now write memory<br>CALL UNLOCK_SEQ ; Perform required unlock sequence<br>BCF NVMCON1,WREN ; Disable writes<br>UNLOCK_SEQ<br>UNLOCK_SEQ<br>MOVLW 55h<br>BCF INTCON,GIE ; Disable interrupts<br>MOVWF NVMCON2 ; Begin unlock sequence<br>MOVLW AAh<br>MOVWF NVMCON2 ; Begin unlock sequence<br>MOVLW AAh<br>MOVWF NVMCON2 ; Begin unlock sequence<br>SFF NVMCON1,WR SF<br>BSF NVMCON1,WR SF<br>BSF NVMCON1,WR SF<br>BSF NVMCON1,WR SF<br>Feturn :   |   | MOVWF  | NVMDATL   | ; Load first data byte   |
| MOVWFNVMDATH; Load second data byteMOVFNVMADRL,WXORLW0x1F; Check if lower bits of address are 00000ANDLW0x1F; and if on last of 32 addressesBTFSCSTATUS,Z; Last of 32 words?GOTOSTART_WRITE; If so, go write latches into memoryCALLUNLOCK_SEQ; If not, go load latchINCFNVMADRL,F; Increment addressGOTOCOPSTART_WRITE; Latch writes complete, now write memoryCALLUNLOCK_SEQ; Perform required unlock sequenceBCFNVMCON1,LWLO; Latch writesBCFNVMCON1,WREN; Disable writesUNLOCK_SEQIntron, GIE; Disable interruptsMOVUWAAhMOVUWFMOVUWAAhSEFMOVWFNVMCON2; Begin unlock sequenceMOVWFNVMCON2; Begin unlock sequenceMOVWFNVMCON2; Begin unlock sequenceMOVWFNVMCON2; Begin unlock sequenceMOVWFNVMCON2; Begin unlock sequenceMOVWFNVMCON1,WR; BSFBSFNVMCON1,WRBSFNVMCON1,WRBSFNVMCON1,WRBSFNVMCON1,WRBSFNVMCON1,WRRSFNVMCON1,WRRSFNVMCON1,WRRSFNVMCON1,WRRSFNVMCON1,WRRSFNVMCON1,WRRSFNVMCON1,WRRSFNVMCON1,WRRSFNVMCON1,WRRSFNVMCON1,WR </td <td></td> <td>MOVIW</td> <td>FSR0++</td> <td></td>   |   | MOVIW  | FSR0++  |  |
| MOVFNVMADRL,WXORLW0x1F; Check if lower bits of address are 00000ANDLW0x1F; and if on last of 32 addressesBTFSCSTATUS,Z; Last of 32 words?GOTOSTART_WRITE; If so, go write latches into memoryCALLUNLOCK_SEQ; If not, go load latchINCFNVMADRL,F; Increment addressGOTOLOOP; Latch writes complete, now write memoryCALLUNLOCK_SEQ; Perform required unlock sequenceBCFNVMCON1,LWLO; Latch writesBCFNVMCON1,WREN; Disable writesUNLOCK_SEQINTCON,GIE; Disable interruptsMOVUW55h;BCFINTCON,GIE; Disable interruptsMOVWFNVMCON2; Begin unlock sequenceMOVWFNVMCON1,WRBSFNVMCON1,WRBSFNVMCON1,WRBSFNVMCON1,WRBSFNVMCON1,WRBSFINTCON,GIEreturn; Unlock sequence complete, re-enable interrupts  |   | MOVWF  | NVMDATH   | ; Load second data byte  |
| XORLW0x1F; Check if lower bits of address are 00000ANDLW0x1F; and if on last of 32 addressesBTFSCSTATUS,Z; Last of 32 words?GOTOSTART_WRITE; If so, go write latches into memoryCALLUNLOCK_SEQ; If not, go load latchINCFNVMADRL,F; Increment addressGOTOLOOPSTART_WRITEBCFNVMCON1,LWLOCALLUNLOCK_SEQ; Perform required unlock sequenceBCFNVMCON1,WREN; Disable writesUNLOCK_SEQINTCON,GIE; Disable interruptsMOVUWAhAhMOVWFNVMCON2; Begin unlock sequenceMOVWFNVMCON1,WR; BSFBSFINTCON,GIE; Unlock sequence complete, re-enable interruptsreturnVINCON,GIE; Unlock sequence complete, re-enable interrupts   |   | MOVF   | NVMADRL,W   |  |
| ANDLW 0x1F ; and if on last of 32 addresses<br>BTFSC STATUS,Z ; Last of 32 words?<br>GOTO START_WRITE ; If so, go write latches into memory<br>CALL UNLOCK_SEQ ; If not, go load latch<br>INCF NVMADRL,F ; Increment address<br>GOTO LOOP<br>START_WRITE<br>BCF NVMCON1,LWLO ; Latch writes complete, now write memory<br>CALL UNLOCK_SEQ ; Perform required unlock sequence<br>BCF NVMCON1,WREN ; Disable writes<br>UNLOCK_SEQ<br>UNLOCK_SEQ<br>MOVLW 55h<br>BCF INTCON,GIE ; Disable interrupts<br>MOVWF NVMCON2 ; Begin unlock sequence<br>MOVLW AAh<br>MOVWF NVMCON2<br>BSF NVMCON1,WR<br>BSF INTCON,GIE ; Unlock sequence complete, re-enable interrupts<br>return   |   | XORLW  | 0x1F  | ; Check if lower bits of address are 00000   |
| BTFSC<br>GOTOSTATUS,Z; Last of 32 words?<br>GOTOGOTOSTART_WRITE; If so, go write latches into memoryCALL<br>INCF<br>GOTOUNLOCK_SEQ; If not, go load latchINCF<br>GOTONVMADRL,F<br>LOOP; Increment addressSTART_WRITEBCF<br>BCFNVMCON1,LWLO; Latch writes complete, now write memoryCALL<br>BCFUNLOCK_SEQ<br>NVMCON1,WREN; Disable writesUNLOCK_SEQ<br>MOVLW55h; Disable interruptsMOVWFNVMCON2; Begin unlock sequenceMOVWFNVMCON2; Begin unlock sequenceMOVWFNVMCON1,WR; Disable interruptsMOVWFNVMCON1; Begin unlock sequenceMOVWFNVMCON1; Unlock sequence complete, re-enable interruptsFeturn; Unlock sequence complete, re-enable interrupts  |   | ANDLW  | 0x1F  | ; and if on last of 32 addresses   |
| GOTO START_WRITE ; If so, go write latches into memory<br>CALL UNLOCK_SEQ ; If not, go load latch<br>INCF NVMADRL,F ; Increment address<br>GOTO LOOP<br>START_WRITE<br>BCF NVMCON1,LWLO ; Latch writes complete, now write memory<br>CALL UNLOCK_SEQ ; Perform required unlock sequence<br>BCF NVMCON1,WREN ; Disable writes<br>UNLOCK_SEQ<br>MOVLW 55h<br>BCF INTCON,GIE ; Disable interrupts<br>MOVLW 55h<br>BCF NVMCON2 ; Begin unlock sequence<br>MOVLW AAh<br>MOVWF NVMCON2<br>BSF NVMCON1,WR<br>BSF NVMCON1,WR<br>BSF INTCON,GIE ; Unlock sequence complete, re-enable interrupts<br>return   |   | BTFSC  | STATUS,Z  | ; Last of 32 words?  |
| CALL UNLOCK_SEQ ; If not, go load latch<br>INCF NVMADRL,F ; Increment address<br>GOTO LOOP<br>START_WRITE<br>BCF NVMCON1,LWLO ; Latch writes complete, now write memory<br>CALL UNLOCK_SEQ ; Perform required unlock sequence<br>BCF NVMCON1,WREN ; Disable writes<br>UNLOCK_SEQ<br>UNLOCK_SEQ<br>MOVLW 55h<br>BCF INTCON,GIE ; Disable interrupts<br>MOVWF NVMCON2 ; Begin unlock sequence<br>MOVLW AAh<br>MOVWF NVMCON2 ; Begin unlock sequence<br>MOVLW AAh<br>MOVWF NVMCON2 ; Begin unlock sequence<br>MOVWF NVMCON2 ; Begin unlock sequence<br>MOVLW AAh<br>MOVWF NVMCON2 ; Unlock sequence complete, re-enable interrupts<br>SF NVMCON1,WR ; Unlock sequence complete, re-enable interrupts   |   | GOTO   | START_WRITE   | ; If so, go write latches into memory  |
| INCF OVMADRL,F ; Increment address<br>GOTO LOOP<br>START_WRITE<br>BCF NVMCON1,LWLO ; Latch writes complete, now write memory<br>CALL UNLOCK_SEQ ; Perform required unlock sequence<br>BCF NVMCON1,WREN ; Disable writes<br>UNLOCK_SEQ<br>UNLOCK_SEQ<br>MOVLW 55h<br>BCF INTCON,GIE ; Disable interrupts<br>MOVWF NVMCON2 ; Begin unlock sequence<br>MOVLW AAh<br>MOVWF NVMCON2 ; Begin unlock sequence<br>BSF NVMCON1,WR<br>BSF INTCON,GIE ; Unlock sequence complete, re-enable interrupts<br>return   |   | CALL   | UNLOCK_SEQ  | ; If not, go load latch  |
| GOTO     LOOP       START_WRITE     BCF     NVMCON1,LWLO     ; Latch writes complete, now write memory<br>CALL       CALL     UNLOCK_SEQ     ; Perform required unlock sequence<br>BCF       MOVLW     55h       BCF     INTCON,GIE     ; Disable interrupts       MOVWF     NVMCON2     ; Begin unlock sequence       MOVLW     AAh       MOVWF     NVMCON2     ; Begin unlock sequence       MOVWF     NVMCON2     ; Begin unlock sequence       MOVWF     NVMCON1,WR     ; Begin unlock sequence       MSF     NVMCON1,WR     ; Begin unlock sequence       MSF     NVMCON1,WR     ; Begin unlock sequence       BSF     INTCON,GIE     ; Unlock sequence complete, re-enable interrupts       return  |   | INCF   | NVMADRL,F   | ; Increment address  |
| START_WRITE         BCF       NVMCON1,LWLO       ; Latch writes complete, now write memory         CALL       UNLOCK_SEQ       ; Perform required unlock sequence         BCF       NVMCON1,WREN       ; Disable writes         UNLOCK_SEQ           MOVLW       55h          BCF       INTCON,GIE       ; Disable interrupts         MOVWF       NVMCON2       ; Begin unlock sequence         MOVWF       NVMCON2       ; Begin unlock sequence         MOVWF       NVMCON1,WR       ; Disable interrupts         BSF       NVMCON1,WR       ; Begin unlock sequence         BSF       INTCON,GIE       ; Unlock sequence complete, re-enable interrupts         return       ; Unlock sequence complete, re-enable interrupts  |   | GOTO   | LOOP  |  |
| BCF       NVMCON1,LWLO       ; Latch writes complete, now write memory         CALL       UNLOCK_SEQ       ; Perform required unlock sequence         BCF       NVMCON1,WREN       ; Disable writes         UNLOCK_SEQ       MOVLW       55h         BCF       INTCON,GIE       ; Disable interrupts         MOVWF       NVMCON2       ; Begin unlock sequence         MOVWF       NVMCON2       ; Begin unlock sequence         MOVWF       NVMCON1,WR       ;         BSF       NVMCON1,WR       ;         BSF       INTCON,GIE       ; Unlock sequence complete, re-enable interrupts         return       .       .   | START_W   | RITE   |   |  |
| CALL UNLOCK_SEQ ; Perform required unlock sequence<br>BCF NVMCON1,WREN ; Disable writes<br>UNLOCK_SEQ<br>MOVLW 55h<br>BCF INTCON,GIE ; Disable interrupts<br>MOVWF NVMCON2 ; Begin unlock sequence<br>MOVLW AAh<br>MOVWF NVMCON2<br>BSF NVMCON1,WR<br>BSF INTCON,GIE ; Unlock sequence complete, re-enable interrupts<br>return   |   | BCF  | NVMCON1,LWLO  | ; Latch writes complete, now write memory  |
| BCF NVMCON1,WREN ; Disable writes<br>UNLOCK_SEQ<br>MOVLW 55h<br>BCF INTCON,GIE ; Disable interrupts<br>MOVWF NVMCON2 ; Begin unlock sequence<br>MOVLW AAh<br>MOVWF NVMCON2<br>BSF NVMCON1,WR<br>BSF INTCON,GIE ; Unlock sequence complete, re-enable interrupts<br>return   |   | CALL   | UNLOCK_SEQ  | ; Perform required unlock sequence   |
| UNLOCK_SEQ<br>MOVLW 55h<br>BCF INTCON,GIE ; Disable interrupts<br>MOVWF NVMCON2 ; Begin unlock sequence<br>MOVLW AAh<br>MOVWF NVMCON2<br>BSF NVMCON1,WR<br>BSF INTCON,GIE ; Unlock sequence complete, re-enable interrupts<br>return  |   | BCF  | NVMCON1,WREN  | ; Disable writes   |
| MOVLW55hBCFINTCON,GIE; Disable interruptsMOVWFNVMCON2; Begin unlock sequenceMOVWFAAh  | UNLOCK_   | SEQ  |   |  |
| BCFINTCON,GIE; Disable interruptsMOVWFNVMCON2; Begin unlock sequenceMOVUWAAh  |   | MOVLW  | 55h   |  |
| MOVWFNVMCON2; Begin unlock sequenceMOVLWAAhMOVWFNVMCON2BSFNVMCON1,WRBSFINTCON,GIEreturn; Unlock sequence complete, re-enable interrupts   |   | BCF  | INTCON,GIE  | ; Disable interrupts   |
| MOVLW     AAh       MOVWF     NVMCON2       BSF     NVMCON1,WR       BSF     INTCON,GIE     ; Unlock sequence complete, re-enable interrupts       return   |   | MOVWF  | NVMCON2   | ; Begin unlock sequence  |
| MOVWF NVMCON2<br>BSF NVMCON1,WR<br>BSF INTCON,GIE ; Unlock sequence complete, re-enable interrupts<br>return  |   | MOVLW  | AAh   |  |
| BSF NVMCON1,WR<br>BSF INTCON,GIE ; Unlock sequence complete, re-enable interrupts<br>return   |   | MOVWF  | NVMCON2   |  |
| BSF INTCON,GIE ; Unlock sequence complete, re-enable interrupts return  |   | BSF  | NVMCON1,WR  |  |
| return  |   | BSF  | INTCON, GIE   | ; Unlock sequence complete, re-enable interrupts   |
|   |   | return   |   |  |

| Name   | Bit 7 | Bit 6 | Bit 5 | Bit 4       | Bit 3 | Bit 2     | Bit 1 | Bit 0 | Register<br>on page |  |
|--------|-------|-------|-------|-------------|-------|-----------|-------|-------|---------------------|--|
| RA4PPS | —     | _     |       | RA4PPS<5:0> |       |           |       |       |                     |  |
| RA5PPS | —     | _     |       |             | RAS   | 5PPS<5:0> |       |       | 215                 |  |
| RA6PPS | _     | _     |       |             | RA6   | )PPS<5:0> |       |       | 215                 |  |
| RA7PPS | _     | _     |       |             | RA7   | 'PPS<5:0> |       |       | 215                 |  |
| RB0PPS | —     | _     |       |             | RBC   | )PPS<5:0> |       |       | 215                 |  |
| RB1PPS | _     | _     |       |             | RB1   | PPS<5:0>  |       |       | 215                 |  |
| RB2PPS | _     | _     |       |             | RB2   | 2PPS<5:0> |       |       | 215                 |  |
| RB3PPS | _     | _     |       | RB3PPS<5:0> |       |           |       |       |                     |  |
| RB4PPS | _     | _     |       | RB4PPS<5:0> |       |           |       |       |                     |  |
| RB5PPS | _     | _     |       | RB5PPS<5:0> |       |           |       |       |                     |  |
| RB6PPS | —     | _     |       | RB6PPS<5:0> |       |           |       |       |                     |  |
| RB7PPS | —     | -     |       | RB7PPS<5:0> |       |           |       |       |                     |  |
| RC0PPS | —     | _     |       | RC0PPS<5:0> |       |           |       |       |                     |  |
| RC1PPS | —     | _     |       | RC1PPS<5:0> |       |           |       |       |                     |  |
| RC2PPS | —     | _     |       | RC2PPS<5:0> |       |           |       |       |                     |  |
| RC3PPS | _     | _     |       | RC3PPS<5:0> |       |           |       |       |                     |  |
| RC4PPS | —     | _     |       |             | RC4   | IPPS<5:0> |       |       | 215                 |  |
| RC5PPS | _     | _     |       | RC5PPS<5:0> |       |           |       |       |                     |  |
| RC6PPS | —     | _     |       |             | RC    | SPPS<5:0> |       |       | 215                 |  |
| RC7PPS | _     | _     |       |             | RC    | 'PPS<5:0> |       |       | 215                 |  |

# TABLE 13-4: SUMMARY OF REGISTERS ASSOCIATED WITH THE PPS MODULE (CONTINUED)

Legend: — = unimplemented, read as '0'. Shaded cells are unused by the PPS module.

| REGISTER 15-7: IOCCP: INTERRUPT-ON-CHANGE PORTC POSITIVE EDGE REGISTE |
|---|
|---|

| R/W-0/0                                   | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0                            | R/W-0/0        | R/W-0/0          | R/W-0/0     |  |
|---|---------|---------|---------|------------------------------------|----------------|------------------|-------------|--|
| IOCCP7                                    | IOCCP6  | IOCCP5  | IOCCP4  | IOCCP3                             | IOCCP2         | IOCCP1           | IOCCP0      |  |
| 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 a                     | at POR and BOI | R/Value at all c | ther Resets |  |

bit 7-0

'1' = Bit is set

IOCCP<7:0>: Interrupt-on-Change PORTC Positive Edge Enable bits

1 = Interrupt-on-Change enabled on the pin for a positive-going edge. IOCCFx bit and IOCIF flag will be set upon detecting an edge.

0 = Interrupt-on-Change disabled for the associated pin

'0' = Bit is cleared

### REGISTER 15-8: IOCCN: INTERRUPT-ON-CHANGE PORTC NEGATIVE EDGE REGISTER

| R/W-0/0 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| IOCCN7  | IOCCN6  | IOCCN5  | IOCCN4  | IOCCN3  | IOCCN2  | IOCCN1  | IOCCN0  |
| bit 7   |         |         |         |         |         |         | bit 0   |

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

bit 7-0

**IOCCN<7:0>:** Interrupt-on-Change PORTC Negative Edge Enable bits

- 1 = Interrupt-on-Change enabled on the pin for a negative-going edge. IOCCFx bit and IOCIF flag will be set upon detecting an edge.
- 0 = Interrupt-on-Change disabled for the associated pin

### REGISTER 15-9: IOCCF: INTERRUPT-ON-CHANGE PORTC FLAG REGISTER

| R/W/HS-0/0 |
|------------|------------|------------|------------|------------|------------|------------|------------|
| IOCCF7     | IOCCF6     | IOCCF5     | IOCCF4     | IOCCF3     | IOCCF2     | IOCCF1     | IOCCF0     |
| bit 7      |            |            |            |            |            |            | bit 0      |

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

bit 7-0

- IOCCF<7:0>: Interrupt-on-Change PORTC Flag bits
- 1 = An enabled change was detected on the associated pin
  - Set when IOCCPx = 1 and a rising edge was detected on RCx, or when IOCCNx = 1 and a falling edge was detected on RCx.
- 0 = No change was detected, or the user cleared the detected change

# 16.0 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 positive input
- Digital-to-Analog Converter (DAC)

The FVR can be enabled by setting the FVREN bit of the FVRCON register.

Note: Fixed Voltage Reference output cannot exceed VDD.

### 16.1 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 23.0** "**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 25.0 "5-Bit Digital-to-Analog Converter (DAC1) Module"** and **Section 18.0 "Comparator Module"** for additional information.

### 16.2 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 FVRRDY bit of the FVRCON register will be set.

#### FIGURE 16-1: VOLTAGE REFERENCE BLOCK DIAGRAM



### **17.4 ADC Acquisition Time**

To ensure accurate temperature measurements, the user must wait at least 200  $\mu$ s after the ADC input multiplexer is connected to the temperature indicator output before the conversion is performed. In addition, the user must wait 200  $\mu$ s between consecutive conversions of the temperature indicator output.

#### TABLE 17-2: SUMMARY OF REGISTERS ASSOCIATED WITH THE TEMPERATURE INDICATOR

| Name   | Bit 7 | Bit 6  | Bit 5 | Bit 4 | Bit 3 | Bit 2  | Bit 1 | Bit 0  | Register<br>on page |
|--------|-------|--------|-------|-------|-------|--------|-------|--------|---------------------|
| FVRCON | FVREN | FVRRDY | TSEN  | TSRNG | CDFVF | R<1:0> | ADFVF | R<1:0> | 234                 |

**Legend:** Shaded cells are unused by the Temperature Indicator module.

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 Table 23-2 for auto-conversion sources.

| ADACT Value | Sour0x1Dce<br>Peripher0x1Dal | Description                                      |
|-------------|------------------------------|--|
| 0x00        | Disabled                     | External Trigger Disabled                        |
| 0x01        | ADACTPPS                     | Pin selected by ADACTPPS                         |
| 0x02        | TMR0                         | Timer0 overflow condition                        |
| 0x03        | TMR1                         | Timer1 overflow condition                        |
| 0x04        | TMR2                         | Match between Timer2<br>postscaled value and PR2 |
| 0x05        | TMR3                         | Timer3 overflow condition                        |
| 0x06        | TMR4                         | Match between Timer4<br>postscaled value and PR4 |
| 0x07        | TMR5                         | Timer5 overflow condition                        |
| 0x08        | TMR6                         | Match between Timer6<br>postscaled value and PR6 |
| 0x09        | SMT1                         | Match between SMT1 and SMT1PR                    |
| 0x0A        | SMT2                         | Match between SMT2 and SMT2PR                    |
| 0x0B        | CCP1                         | CCP1 output                                      |
| 0x0C        | CCP2                         | CCP2 output                                      |
| 0x0D        | CCP3                         | CCP3 output                                      |
| 0x0E        | CCP4                         | CCP4 output                                      |
| 0x0F        | CCP5                         | CCP5 output                                      |
| 0x10        | PWM6                         | PWM6 output                                      |
| 0x11        | PWM7                         | PWM7 output                                      |
| 0x12        | C1                           | Comparator C1 output                             |
| 0x13        | C2                           | Comparator C2 output                             |
| 0x14        | IOC                          | Interrupt-on-change interrupt trigger            |
| 0x15        | CLC1                         | CLC1 output                                      |
| 0x16        | CLC2                         | CLC2 output                                      |
| 0x17        | CLC3                         | CLC3 output                                      |
| 0x18        | CLC4                         | CLC4 output                                      |
| 0x19-0x1B   | Reserved                     | Reserved, do not use                             |
| 0x1C        | ADERR                        | Read of ADERR register                           |
| 0x1D        | ADRESH                       | Read of ADRESH register                          |
| 0x1E        | Reserved                     | Reserved, do not use                             |
| 0x1F        | ADPCH                        | Read of ADPCH register                           |

### TABLE 23-2: ADC AUTO-CONVERSION TABLE

| FIGURE 26-5:                 | CARRIER LOW SYNCHRONIZATION (MDSHSYNC = 0, MDCLSYNC = 1) |
|------------------------------|--|
| Carrier High (CARH)          |  |
| Carrier Low (CARL)           |  |
| Modulator (MOD)              |  |
| MDCHSYNC = 0<br>MDCLSYNC = 1 |  |
| Active Carrier<br>State -    |  |
| FIGURE 26-6:                 | FULL SYNCHRONIZATION (MDSHSYNC = 1, MDCLSYNC = 1)        |
| Carrier High (CARH)          |  |



# 29.0 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 29-1 for a block diagram of Timer2. See Figure 29-2 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 29-1: TIMER2 BLOCK DIAGRAM



# FIGURE 32-12: GATED WINDOWED MEASURE MODE REPEAT ACQUISITION TIMING DIAGRAM



DS40001826A-page 500

PIC16(L)F18854

| PIE8         —         —         SMT2PWAIE         SMT2PRAIE         SMT2IE         SMT1PWAIE         SMT1PRAIE         SMT1IE         123           PIR8         —         —         SMT2PWAIF         SMT2PRAIF         SMT2IF         SMT1PWAIF         SMT1PRAIF         SMT1IF         133           SMT1TMRL         SMT1PWAIF         SMT1PWAIF         SMT1PWAIF         SMT1F         133           SMT1TMRL         SMT1TMR<7:0>          515           SMT1TMRH         SMT1TMR<15:8>         515           SMT1TMRU         SMT1TMR<23:16>         515           SMT1CPRL         SMT1CPR<7:0>         516           SMT1CPRH         SMT1CPR<15:8>         516           SMT1CPRU         SMT1CPR<23:16>         516 | 123<br>133<br>515<br>515<br>515<br>515<br>516 |
|---|---|
| PIR8         —         _         SMT2PWAIF         SMT2PRAIF         SMT2IF         SMT1PWAIF         SMT1PRAIF         SMT1IF         133           SMT1TMRL   | 133<br>515<br>515<br>515<br>515<br>516        |
| SMT1TMRL         SMT1TMR<7:0>         515           SMT1TMRH         SMT1TMR<15:8>         515           SMT1TMRU         SMT1TMR<23:16>         515           SMT1CPRL         SMT1CPR<7:0>         516           SMT1CPRH         SMT1CPR<15:8>         516           SMT1CPRU         SMT1CPR<23:16>         516   | 515<br>515<br>515<br>515<br>516               |
| SMT1TMRH         SMT1TMR<15:8>         515           SMT1TMRU         SMT1TMR<23:16>         515           SMT1CPRL         SMT1CPR<7:0>         516           SMT1CPRH         SMT1CPR<15:8>         516           SMT1CPRU         SMT1CPR<23:16>         516   | 515<br>515<br>516<br>516                      |
| SMT1TMRU         SMT1TMR<23:16>         515           SMT1CPRL         SMT1CPR<7:0>         516           SMT1CPRH         SMT1CPR<15:8>         516           SMT1CPRU         SMT1CPR<23:16>         516  | 515<br>516<br>516                             |
| SMT1CPRL         SMT1CPR<7:0>         516           SMT1CPRH         SMT1CPR<15:8>         516           SMT1CPRU         SMT1CPR<23:16>         516  | 516<br>516                                    |
| SMT1CPRH         SMT1CPR<15:8>         516           SMT1CPRU         SMT1CPR<23:16>         516  | 516   |
| SMT1CPRU SMT1CPR<23:16> 516   |   |
|   | 516   |
| SMT1CPWL SMT1CPW<7:0> 517   | 517   |
| SMT1CPWH         SMT1CPW<15:8>         517  | 517   |
| SMT1CPWU         SMT1CPW<23:16>         517   | 517   |
| SMT1PRL SMT1PR<7:0> 518   | 518   |
| SMT1PRH SMT1PR<15:8> 518  | 518   |
| SMT1PRU SMT1PR<23:16> 518   | 518   |
| SMT1CON0 EN - STP WPOL SPOL CPOL SMT1PS<1:0> 509  | 509   |
| SMT1CON1 SMT1GO REPEAT MODE<3:0> 510  | 510   |
| SMT1STAT CPRUP CPWUP RST TS WS AS 511   | 511   |
| SMT1CLK – – – – CSEL<2:0> 512   | 512   |
| SMT1SIG — — — SSEL<4:0> 514   | 514   |
| SMT1WIN — — — WSEL<4:0> 513   | 513   |
| SMT2TMRL SMT2TMR<7:0> 515   | 515   |
| SMT2TMRH SMT2TMR<15:8> 515  | 515   |
| SMT2TMRU SMT2TMR<23:16> 515   | 515   |
| SMT2CPRL SMT2CPR<7:0> 516   | 516   |
| SMT2CPRH SMT2CPR<15:8> 516  | 516   |
| SMT2CPRU SMT2CPR<23:16> 516   | 516   |
| SMT2CPWL SMT2CPW<7:0> 517   | 517   |
| SMT2CPWH         SMT2CPW<15:8>         517  | 517   |
| SMT2CPWU SMT2CPW<23:16> 517   | 517   |
| SMT2PRL SMT2PR<7:0> 518   | 518   |
| SMT2PRH SMT2PR<15:8> 518  | 518   |
| SMT2PRU SMT2PR<23:16> 518   | 518   |
| SMT2CON0 EN - STP WPOL SPOL CPOL SMT2PS<1:0> 509  | 509   |
| SMT2CON1         SMT2GO         REPEAT         —         —         MODE<3:0>         510  | 510   |
| SMT2STAT CPRUP CPWUP RST TS WS AS 511   | 511   |
| SMT2CLK – – – – CSEL<2:0> 512   | i12   |
| SMT2SIG — — — SSEL<4:0> 514   | 514   |
| SMT2WIN — — — WSEL<4:0> 513   | i13   |

### TABLE 32-3: SUMMARY OF REGISTERS ASSOCIATED WITH SMTx

Legend: - = unimplemented read as '0'. Shaded cells are not used for SMTx module.

# 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