**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 "Embedded - Microcontrollers"

| Details | |
|---|---|
| Product Status | Active |
| Core Processor | PIC |
| Core Size | 8-Bit |
| Speed | 32MHz |
| Connectivity | LINbus, UART/USART |
| Peripherals | Brown-out Detect/Reset, POR, PWM, WDT |
| Number of I/O | 12 |
| Program Memory Size | 7KB (4K x 14) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 512 x 8 |
| Voltage - Supply (Vcc/Vdd) | 1.8V ~ 3.6V |
| Data Converters | A/D 12x10b; D/A 1x5b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 125°C (TA) |
| Mounting Type | Through Hole |
| Package / Case | 20-DIP (0.300", 7.62mm) |
| Supplier Device Package | 20-PDIP |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/pic16lf1578-e-p |

## 3.3 Data Memory Organization

The data memory is partitioned in 32 memory banks with 128 bytes in a bank. Each bank consists of (Figure 3-3):

- 12 core registers
- 20 Special Function Registers (SFR)
- Up to 80 bytes of General Purpose RAM (GPR)
- 16 bytes of common RAM

The active bank is selected by writing the bank number into the Bank Select Register (BSR). Unimplemented memory will read as '0'. All data memory can be accessed either directly (via instructions that use the file registers) or indirectly via the two File Select Registers (FSR). See **Section 3.6 "Indirect Addressing"** for more information.

Data memory uses a 12-bit address. The upper five bits of the address define the Bank address and the lower seven bits select the registers/RAM in that bank.

### 3.3.1 CORE REGISTERS

The core registers contain the registers that directly affect the basic operation. The core registers occupy the first 12 addresses of every data memory bank (addresses x00h/x08h through x0Bh/x8Bh). These registers are listed below in Table 3-2. For detailed information, see Table 3-14.

**TABLE 3-2: CORE REGISTERS**

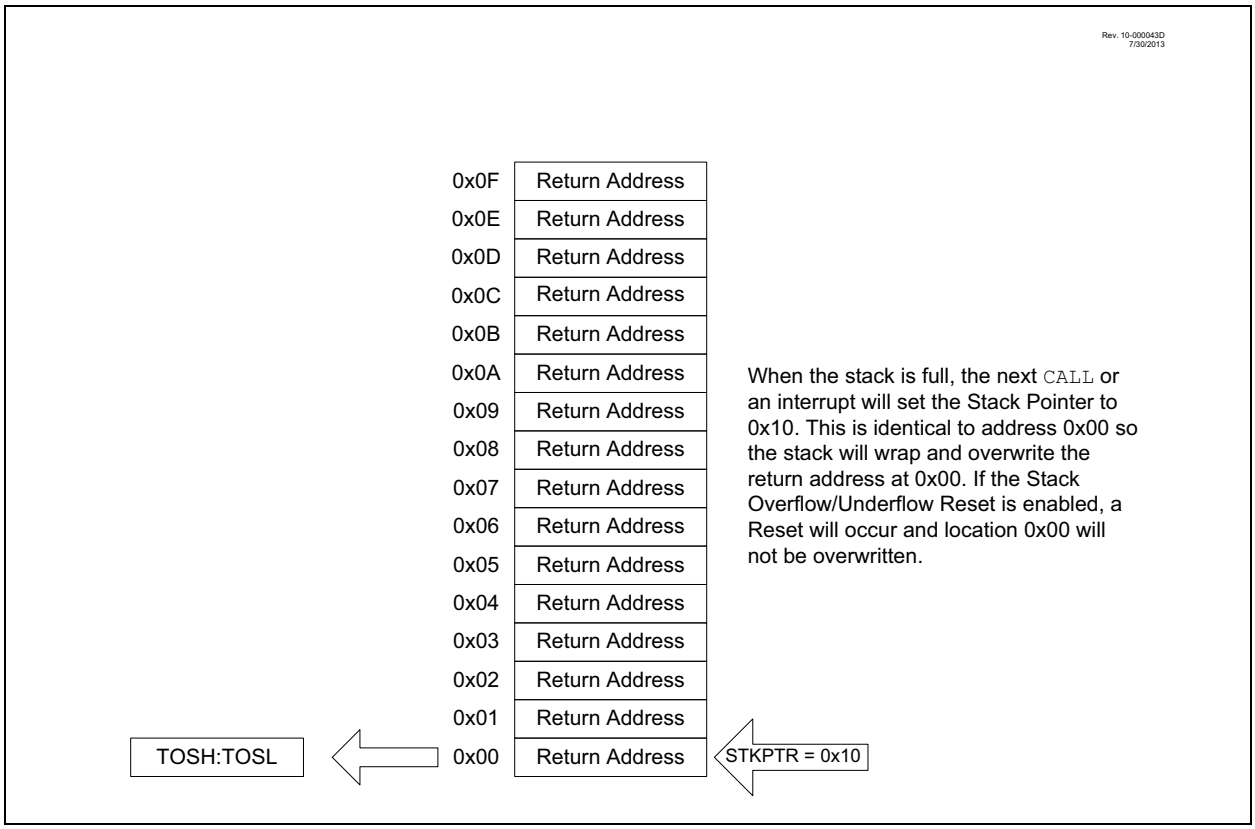
| Addresses | BANKx |
|---|---|
| x00h or x80h | INDF0 |
| x01h or x81h | INDF1 |
| x02h or x82h | PCL |
| x03h or x83h | STATUS |
| x04h or x84h | FSR0L |
| x05h or x85h | FSR0H |
| x06h or x86h | FSR1L |
| x07h or x87h | FSR1H |
| x08h or x88h | BSR |
| x09h or x89h | WREG |
| x0Ah or x8Ah | PCLATH |
| x0Bh or x8Bh | INTCON |

**TABLE 3-8: PIC16(L)F1575/9 MEMORY MAP, BANKS 8-15**

| | BANK 8 | BANK 9 | BANK 10 | BANK 11 | BANK 12 | BANK 13 | BANK 14 | BANK 15 |
|---|---|---|---|---|---|---|---|---|
| 400h | Core Registers (Table 3-2) | 480h Core Registers (Table 3-2) | 500h Core Registers (Table 3-2) | 580h Core Registers (Table 3-2) | 600h Core Registers (Table 3-2) | 680h Core Registers (Table 3-2) | 700h Core Registers (Table 3-2) | 780h Core Registers (Table 3-2) |
| 40Bh | | 48Bh | 50Bh | 58Bh | 60Bh | 68Bh | 70Bh | 78Bh |
| 40Ch | — | 48Ch — | 50Ch — | 58Ch — | 60Ch — | 68Ch — | 70Ch — | 78Ch — |
| 40Dh | — | 48Dh — | 50Dh — | 58Dh — | 60Dh — | 68Dh — | 70Dh — | 78Dh — |
| 40Eh | — | 48Eh — | 50Eh — | 58Eh — | 60Eh — | 68Eh — | 70Eh — | 78Eh — |
| 40Fh | — | 48Fh — | 50Fh — | 58Fh — | 60Fh — | 68Fh — | 70Fh — | 78Fh — |
| 410h | — | 490h — | 510h — | 590h — | 610h — | 690h — | 710h — | 790h — |
| 411h | — | 491h — | 511h — | 591h — | 611h — | 691h CWG1DBR | 711h — | 791h — |
| 412h | — | 492h — | 512h — | 592h — | 612h — | 692h CWG1DBF | 712h — | 792h — |
| 413h | — | 493h — | 513h — | 593h — | 613h — | 693h CWG1CON0 | 713h — | 793h — |
| 414h | — | 494h — | 514h — | 594h — | 614h — | 694h CWG1CON1 | 714h — | 794h — |
| 415h | — | 495h — | 515h — | 595h — | 615h — | 695h CWG1CON2 | 715h — | 795h — |
| 416h | — | 496h — | 516h — | 596h — | 616h — | 696h — | 716h — | 796h — |
| 417h | — | 497h — | 517h — | 597h — | 617h — | 697h — | 717h — | 797h — |
| 418h | — | 498h — | 518h — | 598h — | 618h — | 698h — | 718h — | 798h — |
| 419h | — | 499h — | 519h — | 599h — | 619h — | 699h — | 719h — | 799h — |
| 41Ah | — | 49Ah — | 51Ah — | 59Ah — | 61Ah — | 69Ah — | 71Ah — | 79Ah — |
| 41Bh | — | 49Bh — | 51Bh — | 59Bh — | 61Bh — | 69Bh — | 71Bh — | 79Bh — |
| 41Ch | — | 49Ch — | 51Ch — | 59Ch — | 61Ch — | 69Ch — | 71Ch — | 79Ch — |
| 41Dh | — | 49Dh — | 51Dh — | 59Dh — | 61Dh — | 69Dh — | 71Dh — | 79Dh — |
| 41Eh | — | 49Eh — | 51Eh — | 59Eh — | 61Eh — | 69Eh — | 71Eh — | 79Eh — |
| 41Fh | — | 49Fh — | 51Fh — | 59Fh — | 61Fh — | 69Fh — | 71Fh — | 79Fh — |
| 420h | General Purpose Register 80 Bytes | 4A0h General Purpose Register 80 Bytes | 520h General Purpose Register 80 Bytes | 5A0h General Purpose Register 80 Bytes | 620h General Purpose Register 32 Bytes | 6A0h Unimplemented Read as '0' | 720h Unimplemented Read as '0' | 7A0h Unimplemented Read as '0' |
| 46Fh | | 4EFh | 56Fh | 5EFh | 63Fh | 6EFh | 76Fh | 7EFh |
| 470h | Accesses 70h – 7Fh | 4F0h Accesses 70h – 7Fh | 570h Accesses 70h – 7Fh | 5F0h Accesses 70h – 7Fh | 640h Unimplemented Read as '0' 670h Accesses 70h – 7Fh | 6F0h Accesses 70h – 7Fh | 770h Accesses 70h – 7Fh | 7F0h Accesses 70h – 7Fh |
| 47Fh | | 4FFh | 57Fh | 5FFh | 66Fh | 6FFh | 77Fh | 7FFh |

**Legend:** = Unimplemented data memory locations, read as '0'

**FIGURE 3-8: ACCESSING THE STACK EXAMPLE 4**




### 3.5.2 OVERFLOW/UNDERFLOW RESET

If the STVREN bit in Configuration Words is programmed to '1', the device will be reset if the stack is PUSHed beyond the sixteenth level or POPed beyond the first level, setting the appropriate bits (STKOVF or STKUNF, respectively) in the PCON register.

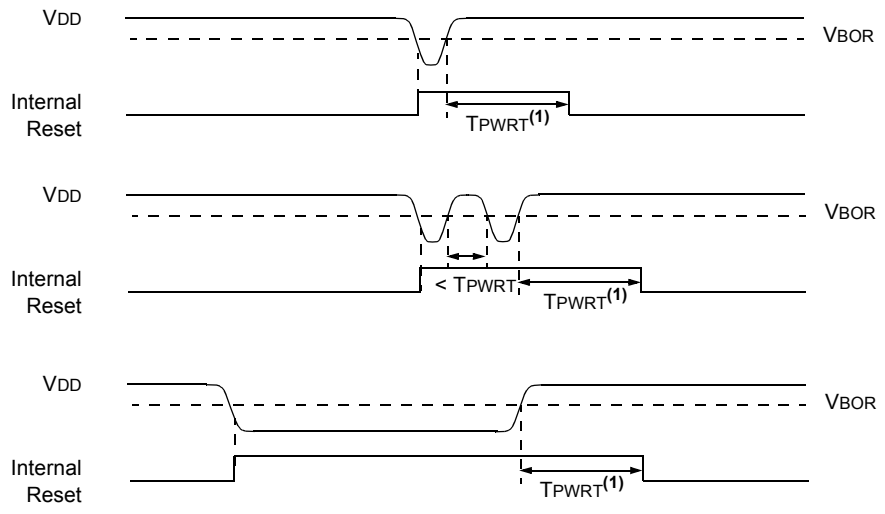
## 3.6 Indirect Addressing

The INDFn registers are not physical registers. Any instruction that accesses an INDFn register actually accesses the register at the address specified by the File Select Registers (FSR). If the FSRn address specifies one of the two INDFn registers, the read will return '0' and the write will not occur (though Status bits may be affected). The FSRn register value is created by the pair FSRnH and FSRnL.

The FSR registers form a 16-bit address that allows an addressing space with 65536 locations. These locations are divided into three memory regions:

- Traditional Data Memory
- Linear Data Memory
- Program Flash Memory

**FIGURE 6-2: BROWN-OUT SITUATIONS**




**Note 1:** TPWRT delay only if $\overline{\text{PWRTE}}$ bit is programmed to '0'.

## 6.3 Register Definitions: BOR Control

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

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

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

bit 7 **SBOREN:** Software Brown-Out Reset Enable bit
If BOREN <1:0> in Configuration Words = 01:
1 = BOR Enabled
0 = BOR Disabled
If BOREN <1:0> in Configuration Words ≠ 01:
SBOREN is read/write, but has no effect on the BOR

bit 6 **BORFS:** Brown-Out Reset Fast Start bit(1)
If BOREN <1:0> = 10 (Disabled in Sleep) or BOREN<1:0> = 01 (Under software control):
1 = Band gap is forced on always (covers sleep/wake-up/operating cases)
0 = Band gap operates normally, and may turn off
If BOREN<1:0> = 11 (Always on) or BOREN<1:0> = 00 (Always off)
BORFS is Read/Write, but has no effect.

bit 5-1 **Unimplemented:** Read as '0'

bit 0 **BORRDY:** Brown-Out Reset Circuit Ready Status bit
1 = The Brown-out Reset circuit is active
0 = The Brown-out Reset circuit is inactive

**Note 1:** BOREN<1:0> bits are located in Configuration Words.

**REGISTER 7-5: PIR1: PERIPHERAL INTERRUPT REQUEST REGISTER 1**

| R/W-0/0 | R/W-0/0 | R-0/0 | R-0/0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| TMR1GIF | ADIF | RCIF | TXIF | — | — | TMR2IF | TMR1IF |
| bit 7 | | | | | | | bit 0 |

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

bit 7 **TMR1GIF:** Timer1 Gate Interrupt Flag bit
1 = Interrupt is pending
0 = Interrupt is not pending

bit 6 **ADIF:** ADC Interrupt Flag bit
1 = Interrupt is pending
0 = Interrupt is not pending

bit 5 **RCIF:** USART Receive Interrupt Flag bit
1 = Interrupt is pending
0 = Interrupt is not pending

bit 4 **TXIF:** USART Transmit Interrupt Flag bit
1 = Interrupt is pending
0 = Interrupt is not pending

bit 3-2 **Unimplemented:** Read as '0'

bit 1 **TMR2IF:** Timer2 to PR2 Interrupt Flag bit
1 = Interrupt is pending
0 = Interrupt is not pending

bit 0 **TMR1IF:** Timer1 Overflow Interrupt Flag bit
1 = Interrupt is pending
0 = Interrupt is not pending

**Note:** Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the Global Interrupt Enable bit, GIE of the INTCON register. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt. The USART RCIF and TXIF bits are read-only.

## 9.1 Independent Clock Source

The WDT derives its time base from the 31 kHz LFINTOSC internal oscillator. Time intervals in this chapter are based on a nominal interval of 1 ms. See **Section 27.0 "Electrical Specifications"** for the LFINTOSC tolerances.

## 9.2 WDT Operating Modes

The Watchdog Timer module has four operating modes controlled by the WDTE<1:0> bits in Configuration Words. See Table 9-1.

### 9.2.1 WDT IS ALWAYS ON

When the WDTE bits of Configuration Words are set to '11', the WDT is always on.

WDT protection is active during Sleep.

### 9.2.2 WDT IS OFF IN SLEEP

When the WDTE bits of Configuration Words are set to '10', the WDT is on, except in Sleep.

WDT protection is not active during Sleep.

### 9.2.3 WDT CONTROLLED BY SOFTWARE

When the WDTE bits of Configuration Words are set to '01', the WDT is controlled by the SWDTEN bit of the WDTCON register.

WDT protection is unchanged by Sleep. See Table 9-1 for more details.

**TABLE 9-1: WDT OPERATING MODES**

| WDTE<1:0> | SWDTEN | Device Mode | WDT Mode |
|---|---|---|---|
| 11 | X | X | Active |
| 10 | X | Awake | Active |
| | | Sleep | Disabled |
| 01 | 1 | X | Active |
| | 0 | X | Disabled |
| 00 | X | X | Disabled |

## 9.3 Time-Out Period

The WDTPS bits of the WDTCON register set the time-out period from 1 ms to 256 seconds (nominal). After a Reset, the default time-out period is two seconds.

## 9.4 Clearing the WDT

The WDT is cleared when any of the following conditions occur:

- Any Reset
- CLRWDT instruction is executed
- Device enters Sleep
- Device wakes up from Sleep
- Oscillator fail
- WDT is disabled
- Oscillator Start-up Timer (OST) is running

See Table 9-2 for more information.

## 9.5 Operation During Sleep

When the device enters Sleep, the WDT is cleared. If the WDT is enabled during Sleep, the WDT resumes counting. When the device exits Sleep, the WDT is cleared again.

The WDT remains clear until the OST, if enabled, completes. See **Section 5.0 "Oscillator Module"** for more information on the OST.

When a WDT time-out occurs while the device is in Sleep, no Reset is generated. Instead, the device wakes up and resumes operation. The $\overline{TO}$ and $\overline{PD}$ bits in the STATUS register are changed to indicate the event. The $\overline{RWDT}$ bit in the PCON register can also be used. See **Section 3.0 "Memory Organization"** for more information.

**TABLE 9-2: WDT CLEARING CONDITIONS**

| Conditions | WDT |
|---|---|
| WDTE<1:0> = 00 | Cleared |
| WDTE<1:0> = 01 and SWDTEN = 0 | |
| WDTE<1:0> = 10 and enter Sleep | |
| CLRWDT Command | |
| Oscillator Fail Detected | |
| Exit Sleep + System Clock = EXTRC, INTOSC, EXTCLK | |
| Change INTOSC divider (IRCF bits) | Unaffected |

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

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|---|---|---|---|---|---|---|---|---|---|
| OSCCON | SPLLEN | IRCF<3:0> | | | | — | SCS<1:0> | | 69 |
| PCON | STKOVF | STKUNF | — | $\overline{\text{RWDT}}$ | $\overline{\text{RMCLR}}$ | $\overline{\text{RI}}$ | $\overline{\text{POR}}$ | $\overline{\text{BOR}}$ | 79 |
| STATUS | — | — | — | $\overline{\text{TO}}$ | $\overline{\text{PD}}$ | Z | DC | C | 23 |
| WDTCON | — | — | WDTPS<4:0> | | | | | SWDTEN | 99 |

**Legend:** x = unknown, u = unchanged, – = 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 on Page |
|---|---|---|---|---|---|---|---|---|---|---|
| CONFIG1 | 13:8 | — | — | — | — | $\overline{\text{CLKOUTEN}}$ | BOREN<1:0> | | — | 56 |
| | 7:0 | $\overline{\text{CP}}$ | MCLRE | $\overline{\text{PWRTE}}$ | WDTE<1:0> | | — | FOSC<1:0> | | |

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

### 10.2.4 WRITING TO FLASH PROGRAM MEMORY

Program memory is programmed using the following steps:

1. Load the address in PMADRH:PMADRL of the row to be programmed.
2. Load each write latch with data.
3. Initiate a programming operation.
4. Repeat steps 1 through 3 until all data is written.

Before writing to program memory, the word(s) to be written must be erased or previously unwritten. Program memory can only be erased one row at a time. No automatic erase occurs upon the initiation of the write.

Program memory can be written one or more words at a time. The maximum number of words written at one time is equal to the number of write latches. See Figure 10-5 (row writes to program memory with 16 write latches) for more details.

The write latches are aligned to the Flash row address boundary defined by the upper 11 bits of PMADRH:PMADRL, (PMADRH<6:0>:PMADRL<7:4>) with the lower 4 bits of PMADRL, (PMADRL<3:0>) determining the write latch being loaded. Write operations do not cross these boundaries. At the completion of a program memory write operation, the data in the write latches is reset to contain 0x3FFF.

The following steps should be completed to load the write latches and program a row of program memory. These steps are divided into two parts. First, each write latch is loaded with data from the PMDATH:PMDATL using the unlock sequence with LWLO = 1. When the last word to be loaded into the write latch is ready, the LWLO bit is cleared and the unlock sequence executed. This initiates the programming operation, writing all the latches into Flash program memory.

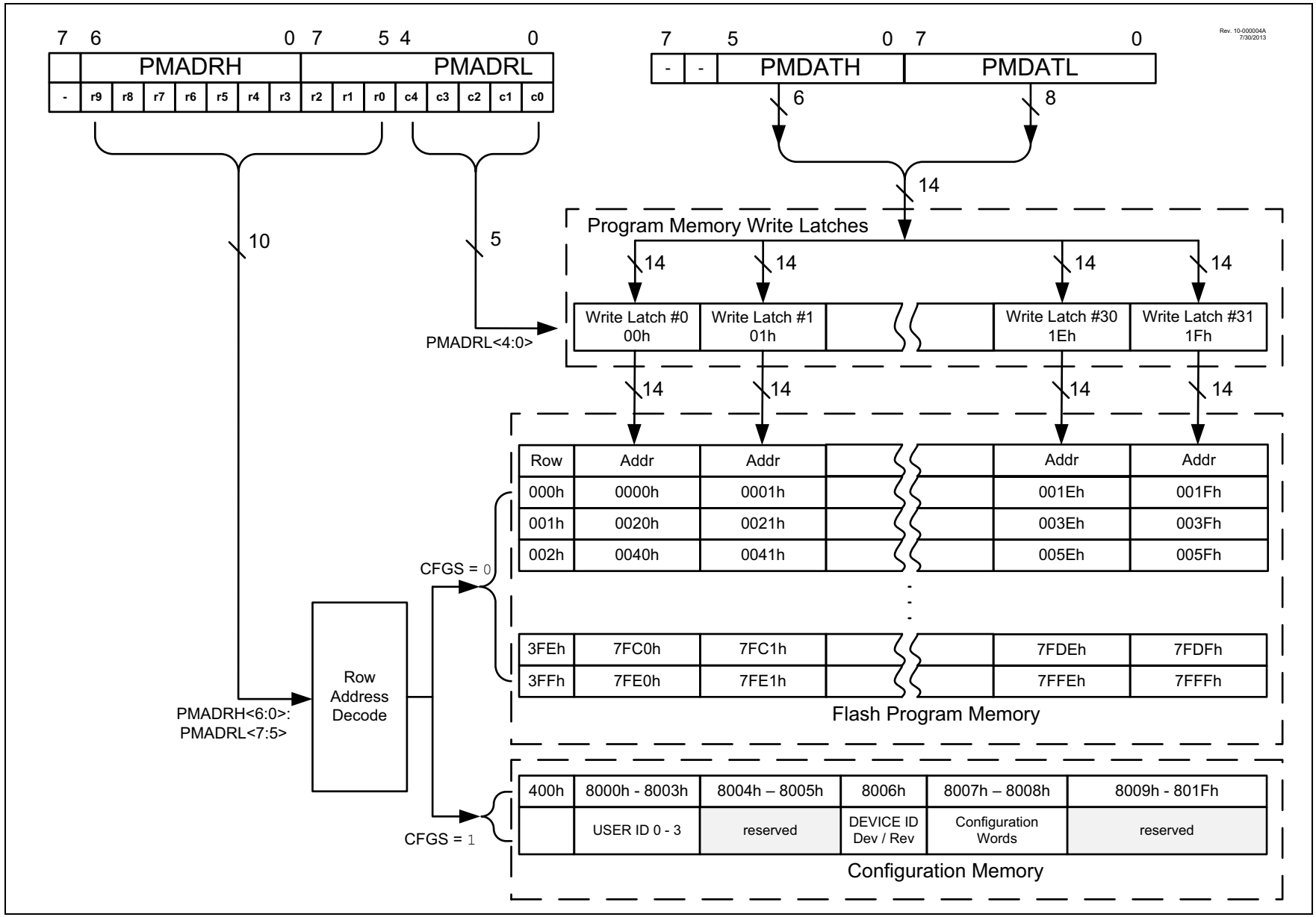
| | |
|---|---|
| **Note:** | The special unlock sequence is required to load a write latch with data or initiate a Flash programming operation. If the unlock sequence is interrupted, writing to the latches or program memory will not be initiated. |

1. Set the WREN bit of the PMCON1 register.
2. Clear the CFGS bit of the PMCON1 register.
3. Set the LWLO bit of the PMCON1 register. When the LWLO bit of the PMCON1 register is '1', the write sequence will only load the write latches and will not initiate the write to Flash program memory.
4. Load the PMADRH:PMADRL register pair with the address of the location to be written.
5. Load the PMDATH:PMDATL register pair with the program memory data to be written.
6. Execute the unlock sequence (**Section 10.2.2 "Flash Memory Unlock Sequence"**). The write latch is now loaded.
7. Increment the PMADRH:PMADRL register pair to point to the next location.
8. Repeat steps 5 through 7 until all but the last write latch has been loaded.
9. Clear the LWLO bit of the PMCON1 register. When the LWLO bit of the PMCON1 register is '0', the write sequence will initiate the write to Flash program memory.
10. Load the PMDATH:PMDATL register pair with the program memory data to be written.
11. Execute the unlock sequence (**Section 10.2.2 "Flash Memory Unlock Sequence"**). The entire program memory latch content is now written to Flash program memory.

| | |
|---|---|
| **Note:** | The program memory write latches are reset to the blank state (0x3FFF) at the completion of every write or erase operation. As a result, it is not necessary to load all the program memory write latches. Unloaded latches will remain in the blank state. |

An example of the complete write sequence is shown in Example 10-3. The initial address is loaded into the PMADRH:PMADRL register pair; the data is loaded using indirect addressing.

**FIGURE 10-5: BLOCK WRITES TO FLASH PROGRAM MEMORY WITH 32 WRITE LATCHES**



**REGISTER 10-6: PMCON2: PROGRAM MEMORY CONTROL 2 REGISTER**

| W-0/0 | W-0/0 | W-0/0 | W-0/0 | W-0/0 | W-0/0 | W-0/0 | W-0/0 |
|---|---|---|---|---|---|---|---|
| Program Memory Control Register 2 | | | | | | | |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| S = Bit can only be set | 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 **Flash Memory Unlock Pattern bits**

To unlock writes, a 55h must be written first, followed by an AAh, before setting the WR bit of the PMCON1 register. The value written to this register is used to unlock the writes. There are specific timing requirements on these writes.

**TABLE 10-3: SUMMARY OF REGISTERS ASSOCIATED WITH FLASH PROGRAM MEMORY**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 86 |
| PMCON1 | —(1) | CFGS | LWLO | FREE | WRERR | WREN | WR | RD | 115 |
| PMCON2 | Program Memory Control Register 2 | | | | | | | | 116 |
| PMADRL | PMADRL<7:0> | | | | | | | | 114 |
| PMADRH | —(1) | PMADRH<6:0> | | | | | | | 114 |
| PMDATL | PMDATL<7:0> | | | | | | | | 114 |
| PMDATH | — | — | PMDATH<5:0> | | | | | | 114 |

**Legend:** — = unimplemented location, read as '0'. Shaded cells are not used by Flash program memory.
**Note 1:** Unimplemented, read as '1'.

**TABLE 10-4: SUMMARY OF CONFIGURATION WORD WITH FLASH PROGRAM MEMORY**

| 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 on Page |
|---|---|---|---|---|---|---|---|---|---|---|
| CONFIG1 | 13:8 | — | — | — | — | $\overline{\text{CLKOUTEN}}$ | BOREN<1:0> | | — | 56 |
| | 7:0 | $\overline{\text{CP}}$ | MCLRE | $\overline{\text{PWRTE}}$ | WDTE<1:0> | | — | FOSC<1:0> | | |
| CONFIG2 | 13:8 | — | — | LVP | $\overline{\text{DEBUG}}$ | $\overline{\text{LPBOR}}$ | BORV | STVREN | PLLEN | 57 |
| | 7:0 | — | — | — | — | — | PPS1WAY | WRT<1:0> | | |

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

## 11.4 Register Definitions: PORTB

### REGISTER 11-9: PORTB: PORTB REGISTER

| R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| RB7 | RB6 | RB5 | RB4 | — | — | — | — |
| bit 7 | | | | | | | bit 0 |

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

bit 7-4 **RB<7:4>**: PORTB General Purpose I/O Pin bits(1)
1 = Port pin is ≥ VIH
0 = Port pin is ≤ VIL

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

**Note 1:** Writes to PORTB are actually written to corresponding LATB register. Reads from PORTB register is return of actual I/O pin values.

### REGISTER 11-10: TRISB: PORTB TRI-STATE REGISTER

| R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| TRISB7 | TRISB6 | TRISB5 | TRISB4 | — | — | — | — |
| bit 7 | | | | | | | bit 0 |

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

bit 7-4 **TRISB<7:4>:** PORTB Tri-State Control bits
1 = PORTB pin configured as an input (tri-stated)
0 = PORTB pin configured as an output

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

### REGISTER 11-11: LATB: PORTB DATA LATCH REGISTER

| R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| LATB7 | LATB6 | LATB5 | LATB4 | — | — | — | — |
| bit 7 | | | | | | | bit 0 |

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

bit 7-4 **LATB<7:4>**: PORTB Output Latch Value bits(1)

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

**Note 1:** Writes to PORTB are actually written to corresponding LATB register. Reads from PORTB register is return of actual I/O pin values.

**TABLE 13-1: SUMMARY OF REGISTERS ASSOCIATED WITH INTERRUPT-ON-CHANGE**

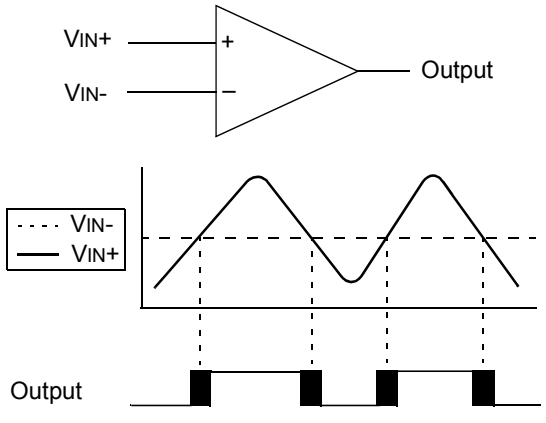
| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|---|---|---|---|---|---|---|---|---|---|
| ANSELA | — | — | — | ANSA4 | — | ANSA2 | ANSA1 | ANSA0 | 121 |
| INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 86 |
| IOCAF | — | — | IOCAF5 | IOCAF4 | IOCAF3 | IOCAF2 | IOCAF1 | IOCAF0 | 143 |
| IOCAN | — | — | IOCAN5 | IOCAN4 | IOCAN3 | IOCAN2 | IOCAN1 | IOCAN0 | 143 |
| IOCAP | — | — | IOCAP5 | IOCAP4 | IOCAP3 | IOCAP2 | IOCAP1 | IOCAP0 | 143 |
| IOCBP(2) | IOCBP7 | IOCBP6 | IOCBP5 | IOCBP4 | — | — | — | — | 144 |
| IOCBN(2) | IOCBN7 | IOCBN6 | IOCBN5 | IOCBN4 | — | — | — | — | 144 |
| IOCBF(2) | IOCBF7 | IOCBF6 | IOCBF5 | IOCBF4 | — | — | — | — | 144 |
| IOCCP | IOCCP7(2) | IOCCP6(2) | IOCCP5 | IOCCP4 | IOCCP3 | IOCCP2 | IOCCP1 | IOCCP0 | 145 |
| IOCCN | IOCCN7(2) | IOCCN6(2) | IOCCN5 | IOCCN4 | IOCCN3 | IOCCN2 | IOCCN1 | IOCCN0 | 145 |
| IOCCF | IOCCF7(2) | IOCCF6(2) | IOCCF5 | IOCCF4 | IOCCF3 | IOCCF2 | IOCCF1 | IOCCF0 | 145 |
| TRISA | — | — | TRISA5 | TRISA4 | —(1) | TRISA2 | TRISA1 | TRISA0 | 120 |
| TRISC | TRISC7(2) | TRISC7(2) | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 131 |

**Legend:** — = unimplemented location, read as '0'. Shaded cells are not used by interrupt-on-change.

**Note 1:** Unimplemented, read as '1'.

**2:** PIC16(L)F1578/9 only.

**FIGURE 18-2: SINGLE COMPARATOR**




**Note:** The black areas of the output of the comparator represents the uncertainty due to input offsets and response time.

## 18.2 Comparator Control

The comparator has two control registers: CMxCON0 and CMxCON1.

The CMxCON0 register (see Register 18-1) contains Control and Status bits for the following:

- Enable
- Output selection
- Output polarity
- Speed/Power selection
- Hysteresis enable
- Output synchronization

The CMxCON1 register (see Register 18-2) contains Control bits for the following:

- Interrupt enable
- Interrupt edge polarity
- Positive input channel selection
- Negative input channel selection

### 18.2.1 COMPARATOR ENABLE

Setting the CxON bit of the CMxCON0 register enables the comparator for operation. Clearing the CxON bit disables the comparator resulting in minimum current consumption.

### 18.2.2 COMPARATOR POSITIVE INPUT SELECTION

Configuring the CxPCH<1:0> bits of the CMxCON1 register directs an internal voltage reference or an analog pin to the non-inverting input of the comparator:

- CxIN+ analog pin
- DAC1_output
- FVR_buffer2
- Vss

See **Section 14.0 "Fixed Voltage Reference (FVR)"** for more information on the Fixed Voltage Reference module.

See **Section 17.0 "5-Bit Digital-to-Analog Converter (DAC) Module"** for more information on the DAC input signal.

Any time the comparator is disabled (CxON = 0), all comparator inputs are disabled.

### 18.2.3 COMPARATOR NEGATIVE INPUT SELECTION

The CxNCH<2:0> bits of the CMxCON0 register direct one of the input sources to the comparator inverting input.

**Note:** To use CxIN+ and CxINx- pins as analog input, the appropriate bits must be set in the ANSEL register and the corresponding TRIS bits must also be set to disable the output drivers.

### 18.2.4 COMPARATOR OUTPUT SELECTION

The output of the comparator can be monitored by reading either the CxOUT bit of the CMxCON0 register or the MCxOUT bit of the CMOUT register. In order to make the output available for an external connection, the following conditions must be true:

- Corresponding TRIS bit must be cleared
- CxON bit of the CMxCON0 register must be set

The synchronous comparator output signal (CxOUT_sync) is available to the following peripheral(s):

- Analog-to-Digital Converter (ADC)
- Timer1

The asynchronous comparator output signal (CxOUT_async) is available to the following peripheral(s):

- Complementary Waveform Generator (CWG)

**Note:** The internal output of the comparator is latched with each instruction cycle. Unless otherwise specified, external outputs are not latched.

**TABLE 20-5: SUMMARY OF REGISTERS ASSOCIATED WITH TIMER1**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|---|---|---|---|---|---|---|---|---|---|
| ANSELA | — | — | — | ANSA4 | — | ANSA2 | ANSA1 | ANSA0 | 121 |
| INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 86 |
| OSCSTAT | — | PLLR | OSTS | HFIOFR | HFIOFL | MFIOFR | LFIOFR | HFIOFS | 70 |
| PIE1 | TMR1GIE | ADIE | RCIE | TXIE | — | — | TMR2IE | TMR1IE | 87 |
| PIR1 | TMR1GIF | ADIF | RCIF | TXIF | — | — | TMR2IF | TMR1IF | 91 |
| TMR1H | Holding Register for the Most Significant Byte of the 16-bit TMR1 Count | | | | | | | | 183* |
| TMR1L | Holding Register for the Least Significant Byte of the 16-bit TMR1 Count | | | | | | | | 183* |
| TRISA | — | — | TRISA5 | TRISA4 | —(1) | TRISA2 | TRISA1 | TRISA0 | 120 |
| T1CON | TMR1CS<1:0> | | T1CKPS<1:0> | | — | $\overline{\text{T1SYNC}}$ | — | TMR1ON | 186 |
| T1GCON | TMR1GE | T1GPOL | T1GTM | T1GSPM | T1GGO/$\overline{\text{DONE}}$ | T1GVAL | T1GSS<1:0> | | 187 |

**Legend:** — = unimplemented location, read as '0'. Shaded cells are not used by the Timer1 module.
* Page provides register information.
**Note 1:** Unimplemented, read as '1'.
**2:** PIC16(L)F1575 only.

## 21.1 Timer2 Operation

The clock input to the Timer2 module is the system instruction clock (FOSC/4).

TMR2 increments from 00h on each clock edge.

A 4-bit counter/prescaler on the clock input allows direct input, divide-by-4 and divide-by-16 prescale options. These options are selected by the prescaler control bits, T2CKPS<1:0> of the T2CON register. The value of TMR2 is compared to that of the Period register, PR2, on each clock cycle. When the two values match, the comparator generates a match signal as the timer output. This signal also resets the value of TMR2 to 00h on the next cycle and drives the output counter/ postscaler (see **Section 21.2 "Timer2 Interrupt"**).

The TMR2 and PR2 registers are both directly readable and writable. The TMR2 register is cleared on any device Reset, whereas the PR2 register initializes to FFh. Both the prescaler and postscaler counters are cleared on the following events:

- a write to the TMR2 register
- a write to the T2CON register
- Power-On Reset (POR)
- Brown-Out Reset (BOR)
- $\overline{\text{MCLR}}$ Reset
- Watchdog Timer (WDT) Reset
- Stack Overflow Reset
- Stack Underflow Reset
- RESET Instruction

| **Note:** | TMR2 is not cleared when T2CON is written. |
|---|---|

## 21.2 Timer2 Interrupt

Timer2 can also generate an optional device interrupt. The Timer2 output signal (T2_match) provides the input for the 4-bit counter/postscaler. This counter generates the TMR2 match interrupt flag which is latched in TMR2IF of the PIR1 register. The interrupt is enabled by setting the TMR2 Match Interrupt Enable bit, TMR2IE of the PIE1 register.

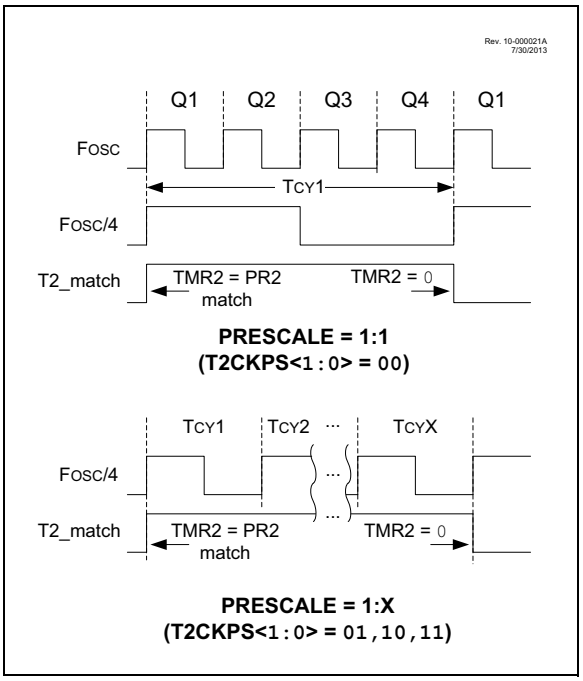
A range of 16 postscale options (from 1:1 through 1:16 inclusive) can be selected with the postscaler control bits, T2OUTPS<3:0>, of the T2CON register.

## 21.3 Timer2 Output

The output of TMR2 is T2_match.

The T2_match signal is synchronous with the system clock. Figure 21-3 shows two examples of the timing of the T2_match signal relative to FOSC and prescale value, T2CKPS<1:0>. The upper diagram illustrates 1:1 prescale timing and the lower diagram, 1:X prescale timing.

**FIGURE 21-3: T2_MATCH TIMING DIAGRAM**




## 21.4 Timer2 Operation During Sleep

Timer2 cannot be operated while the processor is in Sleep mode. The contents of the TMR2 and PR2 registers will remain unchanged while the processor is in Sleep mode.

#### 22.1.2.8 Asynchronous Reception Set-up:

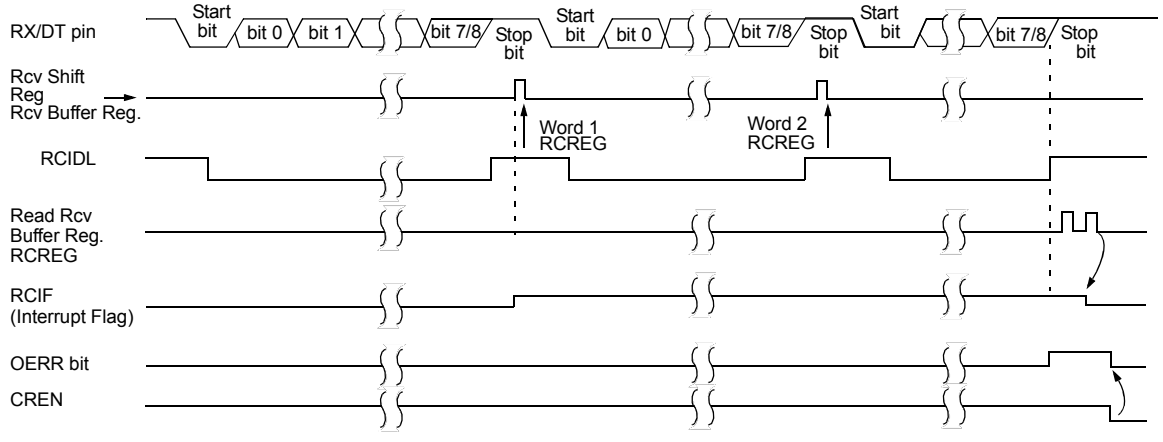
1. Initialize the SPBRGH, SPBRGL register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see **Section 22.4 "EUSART Baud Rate Generator (BRG)"**).
2. Clear the ANSEL bit for the RX pin (if applicable).
3. Enable the serial port by setting the SPEN bit. The SYNC bit must be clear for asynchronous operation.
4. If interrupts are desired, set the RCIE bit of the PIE1 register and the GIE and PEIE bits of the INTCON register.
5. If 9-bit reception is desired, set the RX9 bit.
6. Enable reception by setting the CREN bit.
7. The RCIF interrupt flag bit will be set when a character is transferred from the RSR to the receive buffer. An interrupt will be generated if the RCIE interrupt enable bit was also set.
8. Read the RCSTA register to get the error flags and, if 9-bit data reception is enabled, the ninth data bit.
9. Get the received eight Least Significant data bits from the receive buffer by reading the RCREG register.
10. If an overrun occurred, clear the OERR flag by clearing the CREN receiver enable bit.

#### 22.1.2.9 9-bit Address Detection Mode Set-up

This mode would typically be used in RS-485 systems. To set up an Asynchronous Reception with Address Detect Enable:

1. Initialize the SPBRGH, SPBRGL register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see **Section 22.4 "EUSART Baud Rate Generator (BRG)"**).
2. Clear the ANSEL bit for the RX pin (if applicable).
3. Enable the serial port by setting the SPEN bit. The SYNC bit must be clear for asynchronous operation.
4. If interrupts are desired, set the RCIE bit of the PIE1 register and the GIE and PEIE bits of the INTCON register.
5. Enable 9-bit reception by setting the RX9 bit.
6. Enable address detection by setting the ADDEN bit.
7. Enable reception by setting the CREN bit.
8. The RCIF interrupt flag bit will be set when a character with the ninth bit set is transferred from the RSR to the receive buffer. An interrupt will be generated if the RCIE interrupt enable bit was also set.
9. Read the RCSTA register to get the error flags. The ninth data bit will always be set.
10. Get the received eight Least Significant data bits from the receive buffer by reading the RCREG register. Software determines if this is the device's address.
11. If an overrun occurred, clear the OERR flag by clearing the CREN receiver enable bit.
12. If the device has been addressed, clear the ADDEN bit to allow all received data into the receive buffer and generate interrupts.

**FIGURE 22-5: ASYNCHRONOUS RECEPTION**




**Note:** This timing diagram shows three words appearing on the RX input. The RCREG (receive buffer) is read after the third word, causing the OERR (overrun) bit to be set.

## 23.4 Reload Operation

Four of the PWM module control register pairs and one control bit are double buffered so that all can be updated simultaneously. These include:

- PWMxPHH:PWMxPHL register pair
- PWMxDCH:PWMxDCL register pair
- PWMxPRH:PWMxPRL register pair
- PWMxOFH:PWMxOFL register pair
- OFO control bit

When written to, these registers do not immediately affect the operation of the PWM. By default, writes to these registers will not be loaded into the PWM operating buffer registers until after the arming conditions are met. The arming control has two methods of operation:

- Immediate
- Triggered

The LDT bit of the PWMxLDCON register controls the arming method. Both methods require the LDA bit to be set. All four buffer pairs will load simultaneously at the loading event.

### 23.4.1 IMMEDIATE RELOAD

When the LDT bit is clear then the immediate mode is selected and the buffers will be loaded at the first period event after the LDA bit is set. Immediate reloading is used when a PWM module is operating stand-alone or when the PWM module is operating as a master to other slave PWM modules.

### 23.4.2 TRIGGERED RELOAD

When the LDT bit is set then the Triggered mode is selected and a trigger event is required for the LDA bit to take effect. The trigger source is the buffer load event of one of the other PWM modules in the device. The triggering source is selected by the LDS<1:0> bits of the PWMxLDCON register. The buffers will be loaded at the first period event following the trigger event. Triggered reloading is used when a PWM module is operating as a slave to another PWM and it is necessary to synchronize the buffer reloads in both modules.

| | |
|---|---|
| **Note 1:** | The buffer load operation clears the LDA bit. |
| **2:** | If the LDA bit is set at the same time as PWMxTMR = PWMxPR, the LDA bit is ignored until the next period event. Such is the case when triggered reload is selected and the triggering event occurs simultaneously with the target's period event |

## 23.5 Operation in Sleep Mode

Each PWM module will continue to operate in Sleep mode when either the HFINTOSC or LFINTOSC is selected as the clock source by PWMxCLKCON<1:0>.

## 23.6 Interrupts

Each PWM module has four independent interrupts based on the phase, duty cycle, period, and offset match events. The interrupt flag is set on the rising edge of each of these signals. Refer to Figures 23-8 and 23-12 for detailed timing diagrams of the match signals.

## 24.0 COMPLEMENTARY WAVEFORM GENERATOR (CWG) MODULE

The Complementary Waveform Generator (CWG) produces a complementary waveform with dead-band delay from a selection of input sources.

The CWG module has the following features:

- Selectable dead-band clock source control
- Selectable input sources
- Output enable control
- Output polarity control
- Dead-band control with independent 6-bit rising and falling edge dead-band counters
- Auto-shutdown control with:
  - Selectable shutdown sources
  - Auto-restart enable
  - Auto-shutdown pin override control

## 24.1 Fundamental Operation

The CWG generates two output waveforms from the selected input source.

The off-to-on transition of each output can be delayed from the on-to-off transition of the other output, thereby, creating a time delay immediately where neither output is driven. This is referred to as dead time and is covered in **Section 24.5 "Dead-Band Control"**. A typical operating waveform, with dead band, generated from a single input signal is shown in Figure 24-2.

It may be necessary to guard against the possibility of circuit faults or a feedback event arriving too late or not at all. In this case, the active drive must be terminated before the Fault condition causes damage. This is referred to as auto-shutdown and is covered in **Section 24.9 "Auto-Shutdown Control"**.

## 24.2 Clock Source

The CWG module allows the following clock sources to be selected:

- FOSC (system clock)
- HFINTOSC (16 MHz only)

The clock sources are selected using the G1CS0 bit of the CWGxCON0 register (Register 24-1).

## 24.3 Selectable Input Sources

The CWG generates the output waveforms from the input sources in Table 24-1.

**TABLE 24-1: SELECTABLE INPUT SOURCES**

| Source Peripheral | Signal Name |
|---|---|
| CWG input pin | CWGxIN pin |
| Comparator C1 | C1OUT_sync |
| Comparator C2 | C2OUT_sync |
| PWM1 | PWM1_output |
| PWM2 | PWM2_output |
| PWM3 | PWM3_output |
| PWM4 | PWM4_output |

The input sources are selected using the GxIS<2:0> bits in the CWGxCON1 register (Register 24-2).

## 24.4 Output Control

Immediately after the CWG module is enabled, the complementary drive is configured with both CWGxA and CWGxB drives cleared.

### 24.4.1 POLARITY CONTROL

The polarity of each CWG output can be selected independently. When the output polarity bit is set, the corresponding output is active-high. Clearing the output polarity bit configures the corresponding output as active-low. However, polarity does not affect the override levels. Output polarity is selected with the GxPOLA and GxPOLB bits of the CWGxCON0 register.

## 26.0 INSTRUCTION SET SUMMARY

Each instruction is a 14-bit word containing the operation code (opcode) and all required operands. The opcodes are broken into three broad categories.

- Byte Oriented
- Bit Oriented
- Literal and Control

The literal and control category contains the most varied instruction word format.

Table 26-3 lists the instructions recognized by the MPASM™ assembler.

All instructions are executed within a single instruction cycle, with the following exceptions, which may take two or three cycles:

- Subroutine takes two cycles (CALL, CALLW)
- Returns from interrupts or subroutines take two cycles (RETURN, RETLW, RETFIE)
- Program branching takes two cycles (GOTO, BRA, BRW, BTFSS, BTFSC, DECFSZ, INCSFZ)
- One additional instruction cycle will be used when any instruction references an indirect file register and the file select register is pointing to program memory.

One instruction cycle consists of 4 oscillator cycles; for an oscillator frequency of 4 MHz, this gives a nominal instruction execution rate of 1 MHz.

All instruction examples use the format '0xhh' to represent a hexadecimal number, where 'h' signifies a hexadecimal digit.

### 26.1 Read-Modify-Write Operations

Any instruction that specifies a file register as part of the instruction performs a Read-Modify-Write (R-M-W) operation. The register is read, the data is modified, and the result is stored according to either the instruction, or the destination designator 'd'. A read operation is performed on a register even if the instruction writes to that register.

**TABLE 26-1: OPCODE FIELD DESCRIPTIONS**

| Field | Description |
|---|---|
| f | Register file address (0x00 to 0x7F) |
| W | Working register (accumulator) |
| b | Bit address within an 8-bit file register |
| k | Literal field, constant data or label |
| x | Don't care location (= 0 or 1).<br>The assembler will generate code with x = 0.<br>It is the recommended form of use for compatibility with all Microchip software tools. |
| d | Destination select; d = 0: store result in W,<br>d = 1: store result in file register f.<br>Default is d = 1. |
| n | FSR or INDF number. (0-1) |
| mm | Pre-post increment-decrement mode selection |

**TABLE 26-2: ABBREVIATION DESCRIPTIONS**

| Field | Description |
|---|---|
| PC | Program Counter |
| $\overline{TO}$ | Time-Out bit |
| C | Carry bit |
| DC | Digit Carry bit |
| Z | Zero bit |
| $\overline{PD}$ | Power-Down bit |

## 29.2 MPLAB XC Compilers

The MPLAB XC Compilers are complete ANSI C compilers for all of Microchip's 8, 16, and 32-bit MCU and DSC devices. These compilers provide powerful integration capabilities, superior code optimization and ease of use. MPLAB XC Compilers run on Windows, Linux or MAC OS X.

For easy source level debugging, the compilers provide debug information that is optimized to the MPLAB X IDE.

The free MPLAB XC Compiler editions support all devices and commands, with no time or memory restrictions, and offer sufficient code optimization for most applications.

MPLAB XC Compilers include an assembler, linker and utilities. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. MPLAB XC Compiler uses the assembler to produce its object file. Notable features of the assembler include:

- Support for the entire device instruction set
- Support for fixed-point and floating-point data
- Command-line interface
- Rich directive set
- Flexible macro language
- MPLAB X IDE compatibility

## 29.3 MPASM Assembler

The MPASM Assembler is a full-featured, universal macro assembler for PIC10/12/16/18 MCUs.

The MPASM Assembler generates relocatable object files for the MPLINK Object Linker, Intel® standard HEX files, MAP files to detail memory usage and symbol reference, absolute LST files that contain source lines and generated machine code, and COFF files for debugging.

The MPASM Assembler features include:

- Integration into MPLAB X IDE projects
- User-defined macros to streamline assembly code
- Conditional assembly for multipurpose source files
- Directives that allow complete control over the assembly process

## 29.4 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK Object Linker combines relocatable objects created by the MPASM Assembler. It can link relocatable objects from precompiled libraries, using directives from a linker script.

The MPLIB Object Librarian manages the creation and modification of library files of precompiled code. When a routine from a library is called from a source file, only the modules that contain that routine will be linked in with the application. This allows large libraries to be used efficiently in many different applications.

The object linker/library features include:

- Efficient linking of single libraries instead of many smaller files
- Enhanced code maintainability by grouping related modules together
- Flexible creation of libraries with easy module listing, replacement, deletion and extraction

## 29.5 MPLAB Assembler, Linker and Librarian for Various Device Families

MPLAB Assembler produces relocatable machine code from symbolic assembly language for PIC24, PIC32 and dsPIC DSC devices. MPLAB XC Compiler uses the assembler to produce its object file. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. Notable features of the assembler include:

- Support for the entire device instruction set
- Support for fixed-point and floating-point data
- Command-line interface
- Rich directive set
- Flexible macro language
- MPLAB X IDE compatibility