**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 | - |
| Peripherals | Brown-out Detect/Reset, POR, PWM, WDT |
| Number of I/O | 6 |
| Program Memory Size | 1.75KB (1K x 14) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 128 x 8 |
| Voltage - Supply (Vcc/Vdd) | 1.8V ~ 3.6V |
| Data Converters | A/D 4x10b; D/A 1x5b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 8-TSSOP, 8-MSOP (0.118", 3.00mm Width) |
| Supplier Device Package | 8-MSOP |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/pic12lf1571t-i-ms |

**TABLE 3-10: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)**

| Addr | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Value on All Other Resets |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Bank 4** | | | | | | | | | | | |
| 20Ch | WPUA | — | — | WPUA<5:0> | | | | | | --11 1111 | --11 1111 |
| 20Dh | — | Unimplemented | | | | | | | | — | — |
| 20Eh to 21Fh | — | Unimplemented | | | | | | | | — | — |
| **Bank 5** | | | | | | | | | | | |
| 28Ch | ODCONA | — | — | ODA<5:4> | | — | ODA<2:0> | | | --11 -111 | --11 -111 |
| 28Dh to 29Fh | — | Unimplemented | | | | | | | | — | — |
| **Bank 6** | | | | | | | | | | | |
| 30Ch | SLRCONA | — | — | SLRA<5:4> | | — | SLRA<2:0> | | | --11 -111 | --11 -111 |
| 30Dh to 31Fh | — | Unimplemented | | | | | | | | — | — |
| **Bank 7** | | | | | | | | | | | |
| 38Ch | INLVLA | — | — | INLVLA<5:0> | | | | | | --11 1111 | --11 1111 |
| 38Dh to 390h | — | Unimplemented | | | | | | | | — | — |
| 391h | IOCAP | — | — | IOCAP<5:0> | | | | | | --00 0000 | --00 0000 |
| 392h | IOCAN | — | — | IOCAN<5:0> | | | | | | --00 0000 | --00 0000 |
| 393h | IOCAF | — | — | IOCAF<5:0> | | | | | | --00 0000 | --00 0000 |
| 394h to 39Fh | — | Unimplemented | | | | | | | | — | — |
| **Bank 8** | | | | | | | | | | | |
| 40Ch to 41Fh | — | Unimplemented | | | | | | | | — | — |
| **Bank 9** | | | | | | | | | | | |
| 48Ch to 49Fh | — | Unimplemented | | | | | | | | — | — |

**Legend:** x = unknown; u = unchanged; q = value depends on condition; — = unimplemented; r = reserved. Shaded locations are unimplemented, read as '0'.

**Note 1:** PIC12F1571/2 only.

**2:** PIC12(L)F1572 only.

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

## 5.3 Clock Switching

The system clock source can be switched between external and internal clock sources via software using the System Clock Select (SCSx) bits of the OSCCON register. The following clock sources can be selected using the SCSx bits:

- Default system oscillator determined by FOSCx bits in the Configuration Words
- Timer1 32 kHz crystal oscillator
- Internal Oscillator Block (INTOSC)

### 5.3.1 SYSTEM CLOCK SELECT (SCSx) BITS

The System Clock Select (SCSx) bits of the OSCCON register select the system clock source that is used for the CPU and peripherals.

- When the SCSx bits of the OSCCON register = 00, the system clock source is determined by the value of the FOSC<1:0> bits in the Configuration Words.
- When the SCSx bits of the OSCCON register = 01, the system clock source is the Timer1 oscillator.
- When the SCSx bits of the OSCCON register = 1x, the system clock source is chosen by the internal oscillator frequency selected by the IRCF<3:0> bits of the OSCCON register. After a Reset, the SCSx bits of the OSCCON register are always cleared.

**Note:** Any automatic clock switch does not update the SCSx bits of the OSCCON register. The user can monitor the OSTS bit of the OSCSTAT register to determine the current system clock source.

When switching between clock sources, a delay is required to allow the new clock to stabilize. These oscillator delays are shown in Table 5-1.

## 5.4 Clock Switching Before Sleep

When clock switching from an old clock to a new clock is requested, just prior to entering Sleep mode, it is necessary to confirm that the switch is complete before the SLEEP instruction is executed. Failure to do so may result in an incomplete switch and consequential loss of the system clock altogether. Clock switching is confirmed by monitoring the clock status bits in the OSCSTAT register. Switch confirmation can be accomplished by sensing that the ready bit for the new clock is set or the ready bit for the old clock is cleared. For example, when switching between the internal oscillator with the PLL and the internal oscillator without the PLL, monitor the PLLR bit. When PLLR is set, the switch to 32 MHz operation is complete. Conversely, when PLLR is cleared, the switch from 32 MHz operation to the selected internal clock is complete.

**TABLE 5-1: OSCILLATOR SWITCHING DELAYS**

| Switch From | Switch To | Frequency | Oscillator Delay |
|---|---|---|---|
| Sleep/POR | LFINTOSC(1)<br>MFINTOSC(1)<br>HFINTOSC(1) | 31 kHz<br>31.25 kHz-500 kHz<br>31.25 kHz-16 MHz | Oscillator Warm-up Delay (TWARM)(2) |
| Sleep/POR | EC(1) | DC – 32 MHz | 2 cycles |
| LFINTOSC | EC(1) | DC – 32 MHz | 1 cycle of each |
| Any Clock Source | MFINTOSC(1)<br>HFINTOSC(1) | 31.25 kHz-500 kHz<br>31.25 kHz-16 MHz | 2 μs (approx.) |
| Any Clock Source | LFINTOSC(1) | 31 kHz | 1 cycle of each |
| PLL Inactive | PLL Active | 16-32 MHz | 2 ms (approx.) |

**Note 1:** PLL inactive.

**2:** See **Section 26.0 "Electrical Specifications"**.

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

The Low-Power Brown-out Reset (LPBOR) operates like the BOR to detect low-voltage conditions on the VDD pin. When too low of a voltage is detected, the device is held in Reset. When this occurs, a register bit (BOR) is changed to indicate that a BOR Reset has occurred. The BOR bit in PCON is used for both BOR and the LPBOR. Refer to Register 6-2.

The LPBOR Voltage Threshold (VLPBOR) has a wider tolerance than the BOR (VBOR), but requires much less current (LPBOR current) to operate. The LPBOR is intended for use when the BOR is configured as disabled (BOREN<1:0> = 00) or disabled in Sleep mode (BOREN<1:0> = 10).

Refer to Figure 6-1 to see how the LPBOR interacts with other modules.

### 6.4.1 ENABLING LPBOR

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

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

The $\overline{\text{MCLR}}$ is an optional external input that can reset the device. The $\overline{\text{MCLR}}$ function is controlled by the MCLRE and LVP bits of the Configuration Words (Table 6-2).

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

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

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

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

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

| **Note:** | A Reset does not drive the $\overline{\text{MCLR}}$ pin low. |
|---|---|

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

When $\overline{\text{MCLR}}$ is disabled, the pin functions as a general purpose input and the internal weak pull-up is under software control. See **Section 11.3 "PORTA Registers"** for more information.

## 6.6 Watchdog Timer (WDT) Reset

The Watchdog Timer generates a Reset if the firmware does not issue a CLRWDT instruction within the time-out period. The $\overline{\text{TO}}$ and $\overline{\text{PD}}$ bits in the STATUS register are changed to indicate the WDT Reset. See **Section 9.0 "Watchdog Timer (WDT)"** for more information.

## 6.7 RESET Instruction

A RESET instruction will cause a device Reset. The $\overline{\text{RI}}$ bit in the PCON register will be set to '0'. See Table 6-4 for default conditions after a RESET instruction has occurred.

## 6.8 Stack Overflow/Underflow Reset

The device can reset when the Stack overflows or underflows. The STKOVF or STKUNF bits of the PCON register indicate the Reset condition. These Resets are enabled by setting the STVREN bit in the Configuration Words. See **Section 3.5.2 "Overflow/Underflow Reset"** for more information.

## 6.9 Programming Mode Exit

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

## 6.10 Power-up Timer

The Power-up Timer optionally delays device execution after a BOR or POR event. This timer is typically used to allow VDD to stabilize before allowing the device to start running.

The Power-up Timer is controlled by the $\overline{\text{PWRTE}}$ bit of the Configuration Words.

## 6.11 Start-up Sequence

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

1. Power-up Timer runs to completion (if enabled).
2. $\overline{\text{MCLR}}$ must be released (if enabled).

The total time-out will vary based on oscillator configuration and Power-up Timer configuration. See **Section 5.0 "Oscillator Module"** for more information.

The Power-up Timer runs independently of a $\overline{\text{MCLR}}$ Reset. If $\overline{\text{MCLR}}$ is kept low long enough, the Power-up Timer will expire. Upon bringing $\overline{\text{MCLR}}$ high, the device will begin execution after 10 FOSC cycles (see Figure 6-3). This is useful for testing purposes or to synchronize more than one device operating in parallel.

See Table 10-1 for erase row size and the number of write latches for Flash program memory.

**TABLE 10-1: FLASH MEMORY ORGANIZATION BY DEVICE**

| Device | Row Erase (words) | Write Latches (words) |
|---|---|---|
| PIC12(L)F1571 | 16 | 16 |
| PIC12(L)F1572 | | |

### 10.2.1 READING THE FLASH PROGRAM MEMORY

To read a program memory location, the user must:

1. Write the desired address to the PMADRH:PMADRL register pair.
2. Clear the CFGS bit of the PMCON1 register.
3. Then, set control bit, RD, of the PMCON1 register.

Once the read control bit is set, the program memory Flash controller will use the second instruction cycle to read the data. This causes the second instruction immediately following the "BSF PMCON1,RD" instruction to be ignored. The data is available in the very next cycle in the PMDATH:PMDATL register pair; therefore, it can be read as two bytes in the following instructions.

The PMDATH:PMDATL register pair will hold this value until another read or until it is written to by the user.

| | |
|---|---|
| **Note:** | The two instructions following a program memory read are required to be NOPs. This prevents the user from executing a 2-cycle instruction on the next instruction after the RD bit is set. |

**FIGURE 10-1: FLASH PROGRAM MEMORY READ FLOWCHART**

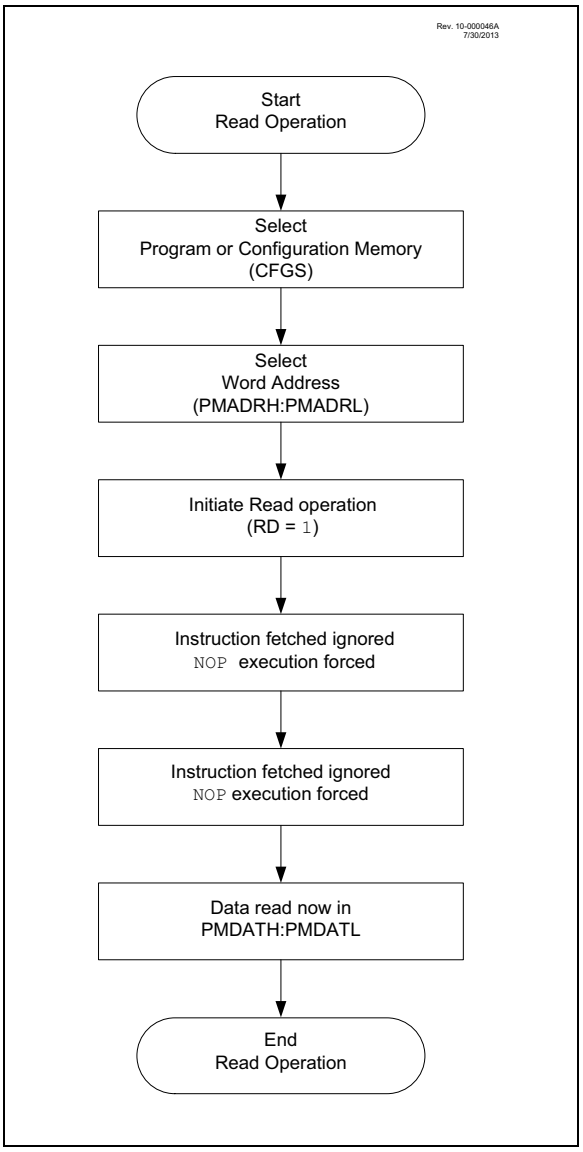
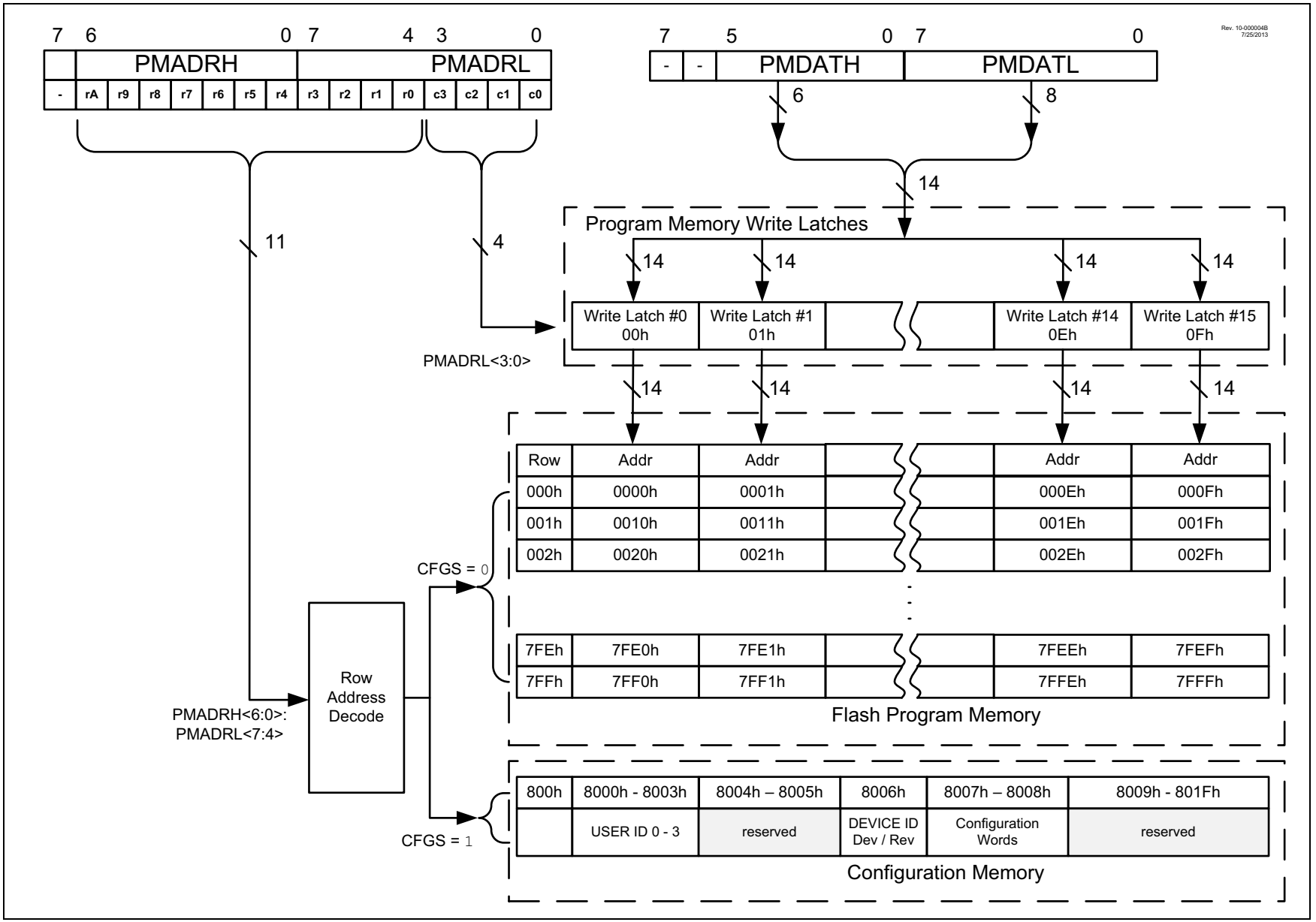


**FIGURE 10-5: BLOCK WRITES TO FLASH PROGRAM MEMORY WITH 16 WRITE LATCHES**



## 10.4 User ID, Device ID and Configuration Word Access

Instead of accessing program memory, the User IDs, Device ID/Revision ID and Configuration Words can be accessed when CFGS = 1 in the PMCON1 register. This is the region that would be pointed to by PC<15> = 1, but not all addresses are accessible. Different access may exist for reads and writes. Refer to Table 10-2.

When read access is initiated on an address outside the parameters listed in Table 10-2, the PMDATH:PMDATL register pair is cleared, reading back '0's.

**TABLE 10-2: USER ID, DEVICE ID AND CONFIGURATION WORD ACCESS (CFGS = 1)**

| Address | Function | Read Access | Write Access |
|---|---|---|---|
| 8000h-8003h | User IDs | Yes | Yes |
| 8006h/8005h | Device ID/Revision ID | Yes | No |
| 8007h-8008h | Configuration Words 1 and 2 | Yes | No |

**EXAMPLE 10-4: CONFIGURATION WORD AND DEVICE ID ACCESS**

```
* This code block will read 1 word of program memory at the memory address:
*   PROG_ADDR_LO (must be 00h-08h) data will be returned in the variables;
*   PROG_DATA_HI, PROG_DATA_LO

    BANKSEL  PMADRL           ; Select correct Bank
    MOVLW    PROG_ADDR_LO     ;
    MOVWF    PMADRL           ; Store LSB of address
    CLRF     PMADRH           ; Clear MSB of address

    BSF      PMCON1,CFGS      ; Select Configuration Space
    BCF      INTCON,GIE       ; Disable interrupts
    BSF      PMCON1,RD        ; Initiate read
    NOP                       ; Executed (See Figure 10-2)
    NOP                       ; Ignored (See Figure 10-2)
    BSF      INTCON,GIE       ; Restore interrupts

    MOVF     PMDATL,W         ; Get LSB of word
    MOVWF    PROG_DATA_LO     ; Store in user location
    MOVF     PMDATH,W         ; Get MSB of word
    MOVWF    PROG_DATA_HI     ; Store in user location
```

## 11.4 Register Definitions: PORTA

**REGISTER 11-2: PORTA: PORTA REGISTER**

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

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

bit 7-6 **Unimplemented:** Read as '0'

bit 5-0 **RA<5:0>**: PORTA I/O Value bits(1)
1 = Port pin is ≥ VIH
0 = Port pin is ≤ VIL

**Note 1:** Writes to PORTA are actually written to corresponding LATA register. Reads from the PORTA register are the return of actual I/O pin values.

**REGISTER 11-3: TRISA: PORTA TRI-STATE REGISTER**

| U-0 | U-0 | R/W-1/1 | R/W-1/1 | U-1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
|---|---|---|---|---|---|---|---|
| — | — | TRISA<5:4> | | —(1) | TRISA<2:0> | | |
| bit 7 | | | | | | | bit 0 |

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

bit 7-6 **Unimplemented:** Read as '0'

bit 5-4 **TRISA<5:4>:** PORTA Tri-State Control bits
1 = PORTA pin configured as an input (tri-stated)
0 = PORTA pin configured as an output

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

bit 2-0 **TRISA<2:0>:** PORTA Tri-State Control bits
1 = PORTA pin configured as an input (tri-stated)
0 = PORTA pin configured as an output

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

### 15.1.5 INTERRUPTS

The ADC module allows for the ability to generate an interrupt upon completion of an Analog-to-Digital conversion. The ADC Interrupt Flag is the ADIF bit in the PIR1 register. The ADC Interrupt Enable is the ADIE bit in the PIE1 register. The ADIF bit must be cleared in software.

| | |
|---|---|
| **Note 1:** | The ADIF bit is set at the completion of every conversion, regardless of whether or not the ADC interrupt is enabled. |
| **2:** | The ADC operates during Sleep only when the FRC oscillator is selected. |

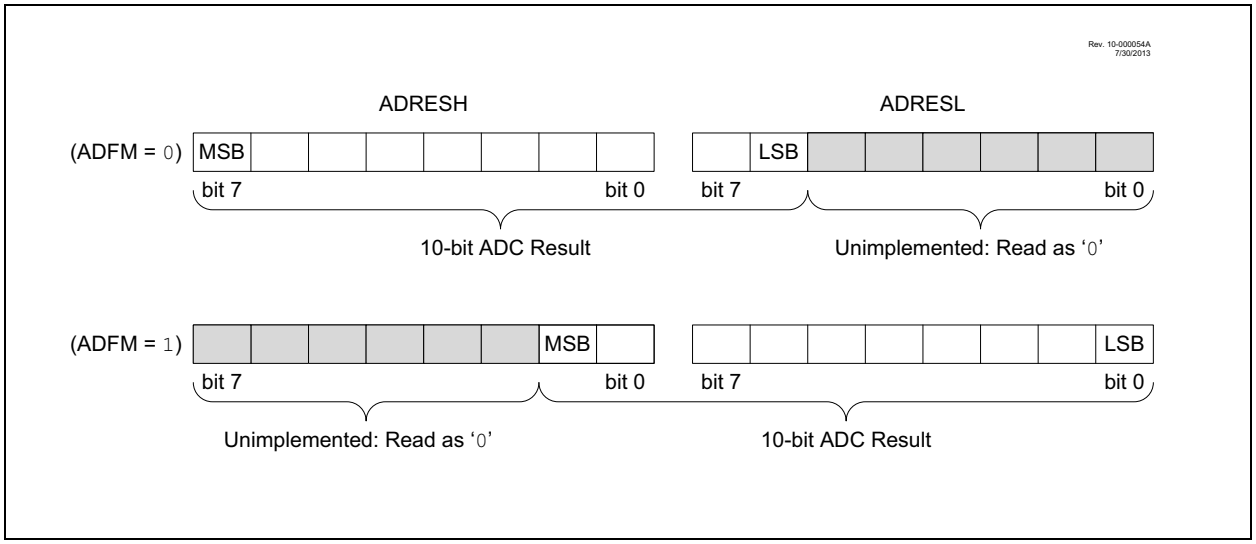
This interrupt can be generated while the device is operating or while in Sleep. If the device is in Sleep, the interrupt will wake-up the device. Upon waking from Sleep, the next instruction following the SLEEP instruction is always executed. If the user is attempting to wake-up from Sleep and resume in-line code execution, the ADIE bit of the PIE1 register and the PEIE bit of the INTCON register must both be set, and the GIE bit of the INTCON register must be cleared. If all three of these bits are set, the execution will switch to the Interrupt Service Routine.

### 15.1.6 RESULT FORMATTING

The 10-bit ADC conversion result can be supplied in two formats, left justified or right justified. The ADFM bit of the ADCON1 register controls the output format.

Figure 15-3 shows the two output formats.

**FIGURE 15-3: 10-BIT ADC CONVERSION RESULT FORMAT**



## 19.1 Timer1 Operation

The Timer1 module is a 16-bit incrementing counter which is accessed through the TMR1H:TMR1L register pair. Writes to TMR1H or TMR1L directly update the counter.

When used with an internal clock source, the module is a timer and increments on every instruction cycle. When used with an external clock source, the module can be used as either a timer or counter and increments on every selected edge of the external source.

Timer1 is enabled by configuring the TMR1ON and TMR1GE bits in the T1CON and T1GCON registers, respectively. Table 19-1 displays the Timer1 enable selections.

**TABLE 19-1: TIMER1 ENABLE SELECTIONS**

| TMR1ON | TMR1GE | Timer1 Operation |
|---|---|---|
| 0 | 0 | Off |
| 0 | 1 | Off |
| 1 | 0 | Always On |
| 1 | 1 | Count Enabled |

## 19.2 Clock Source Selection

The TMR1CS<1:0> bits of the T1CON register are used to select the clock source for Timer1. Table 19-2 displays the clock source selections.

### 19.2.1 INTERNAL CLOCK SOURCE

When the internal clock source is selected, the TMR1H:TMR1L register pair will increment on multiples of FOSC, as determined by the Timer1 prescaler.

When the FOSC internal clock source is selected, the Timer1 register value will increment by four counts every instruction clock cycle. Due to this condition, a 2 LSB error in resolution will occur when reading the Timer1 value. To utilize the full resolution of Timer1, an asynchronous input signal must be used to gate the Timer1 clock input.

The following asynchronous sources may be used:

- Asynchronous event on the T1G pin to Timer1 gate
- C1 or C2 comparator input to Timer1 gate

### 19.2.2 EXTERNAL CLOCK SOURCE

When the external clock source is selected, the Timer1 module may work as a timer or a counter.

When enabled to count, Timer1 is incremented on the rising edge of the external clock input T1CKI. The external clock source can be synchronized to the microcontroller system clock or it can run asynchronously.

**Note:** In Counter mode, a falling edge must be registered by the counter prior to the first incrementing rising edge after any one or more of the following conditions:

- Timer1 enabled after POR
- Write to TMR1H or TMR1L
- Timer1 is disabled
- Timer1 is disabled (TMR1ON = 0) when T1CKI is high, then Timer1 is enabled (TMR1ON = 1) when T1CKI is low

**TABLE 19-2: CLOCK SOURCE SELECTIONS**

| TMR1CS<1:0> | T1OSCEN(1) | Clock Source |
|---|---|---|
| 11 | x | LFINTOSC |
| 10 | x | External Clocking on T1CKI Pin |
| 01 | x | System Clock (FOSC) |
| 00 | x | Instruction Clock (FOSC/4) |

**Note 1:** T1OSCEN is not available for these devices.

**REGISTER 21-3: BAUDCON: BAUD RATE CONTROL REGISTER**

| R-0/0 | R-1/1 | U-0 | R/W-0/0 | R/W-0/0 | U-0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN |
| bit 7 | | | | | | | bit 0 |

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

bit 7 **ABDOVF:** Auto-Baud Detect Overflow bit
Asynchronous mode:
1 = Auto-baud timer overflowed
0 = Auto-baud timer did not overflow
Synchronous mode:
Don't care.

bit 6 **RCIDL**: Receive Idle Flag bit
Asynchronous mode:
1 = Receiver is Idle
0 = Start bit has been received and the receiver is receiving
Synchronous mode:
Don't care.

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

bit 4 **SCKP**: Synchronous Clock Polarity Select bit
Asynchronous mode:
1 = Transmits inverted data to the TX/CK pin
0 = Transmits non-inverted data to the TX/CK pin
Synchronous mode:
1 = Data is clocked on rising edge of the clock
0 = Data is clocked on falling edge of the clock

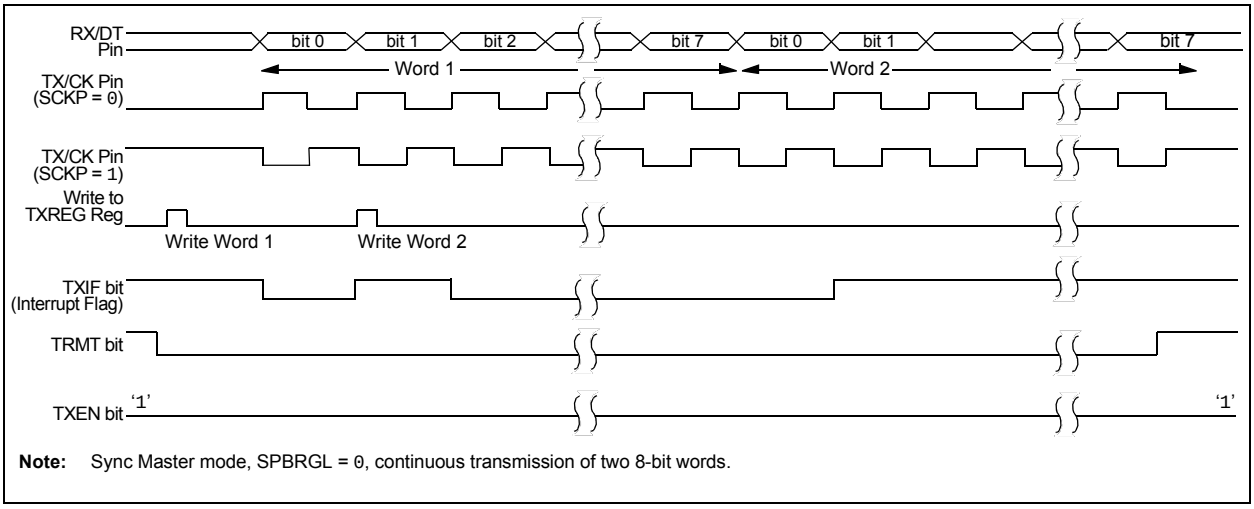
bit 3 **BRG16:** 16-Bit Baud Rate Generator bit
1 = 16-bit Baud Rate Generator is used
0 = 8-bit Baud Rate Generator is used

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

bit 1 **WUE:** Wake-up Enable bit
Asynchronous mode:
1 = Receiver is waiting for a falling edge; no character will be received, RCIF bit will be set, WUE will automatically clear after RCIF is set
0 = Receiver is operating normally
Synchronous mode:
Don't care

bit 0 **ABDEN**: A.uto-Baud Detect Enable bit
Asynchronous mode:
1 = Auto-Baud Detect mode is enabled (clears when auto-baud is complete)
0 = Auto-Baud Detect mode is disabled
Synchronous mode:
Don't care.

**FIGURE 21-10: SYNCHRONOUS TRANSMISSION**




**Note:** Sync Master mode, SPBRGL = 0, continuous transmission of two 8-bit words.

**FIGURE 21-11: SYNCHRONOUS TRANSMISSION (THROUGH TXEN)**

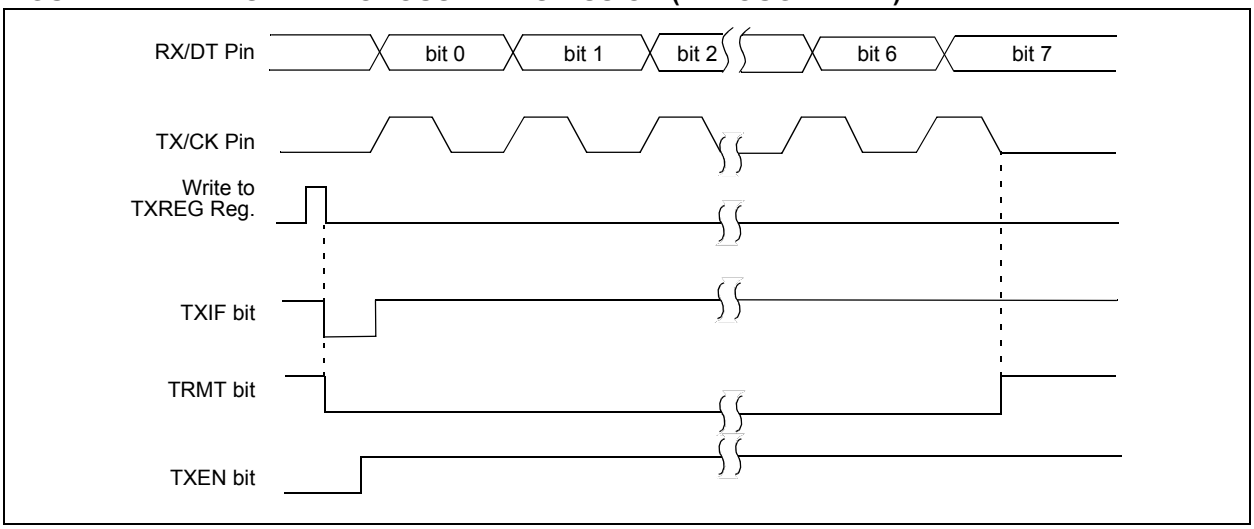



**TABLE 21-7: SUMMARY OF REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER TRANSMISSION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|---|---|---|---|---|---|---|---|---|---|
| BAUDCON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 186 |
| INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 74 |
| PIE1 | TMR1GIE | ADIE | RCIE(1) | TXIE(1) | — | — | TMR2IE | TMR1IE | 75 |
| PIR1 | TMR1GIF | ADIF | RCIF(1) | TXIF(1) | — | — | TMR2IF | TMR1IF | 78 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 185 |
| SPBRGL | BRG<7:0> | | | | | | | | 187* |
| SPBRGH | BRG<15:8> | | | | | | | | 187* |
| TXREG | EUSART Transmit Data Register | | | | | | | | 177* |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 184 |

**Legend:** — = unimplemented location, read as '0'. Shaded cells are not used for synchronous master transmission.

* Page provides register information.

**Note 1:** PIC12(L)F1572 only.

### 21.5.2 SYNCHRONOUS SLAVE MODE

The following bits are used to configure the EUSART for synchronous slave operation:

- SYNC = 1
- CSRC = 0
- SREN = 0 (for transmit); SREN = 1 (for receive)
- CREN = 0 (for transmit); CREN = 1 (for receive)
- SPEN = 1

Setting the SYNC bit of the TXSTA register configures the device for synchronous operation. Clearing the CSRC bit of the TXSTA register configures the device as a slave. Clearing the SREN and CREN bits of the RCSTA register ensures that the device is in Transmit mode; otherwise, the device will be configured to receive. Setting the SPEN bit of the RCSTA register enables the EUSART.

#### 21.5.2.1 EUSART Synchronous Slave Transmit

The operation of the Synchronous Master and Slave modes is identical (see **Section 21.5.1.3 "Synchronous Master Transmission")**, except in the case of Sleep mode.

If two words are written to the TXREG and then the SLEEP instruction is executed, the following will occur:

1. The first character will immediately transfer to the TSR register and transmit.
2. The second word will remain in the TXREG register.
3. The TXIF bit will not be set.
4. After the first character has been shifted out of TSR, the TXREG register will transfer the second character to the TSR and the TXIF bit will now be set.
5. If the PEIE and TXIE bits are set, the interrupt will wake the device from Sleep and execute the next instruction. If the GIE bit is also set, the program will call the Interrupt Service Routine.

#### 21.5.2.2 Synchronous Slave Transmission Setup

1. Set the SYNC and SPEN bits, and clear the CSRC bit.
2. Clear the ANSELx bit for the CK pin (if applicable).
3. Clear the CREN and SREN bits.
4. If interrupts are desired, set the TXIE bit of the PIE1 register and the GIE and PEIE bits of the INTCON register.
5. If 9-bit transmission is desired, set the TX9 bit.
6. Enable transmission by setting the TXEN bit.
7. If 9-bit transmission is selected, insert the Most Significant bit into the TX9D bit.
8. Start transmission by writing the Least Significant eight bits to the TXREG register.

**TABLE 21-9: SUMMARY OF REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE TRANSMISSION**

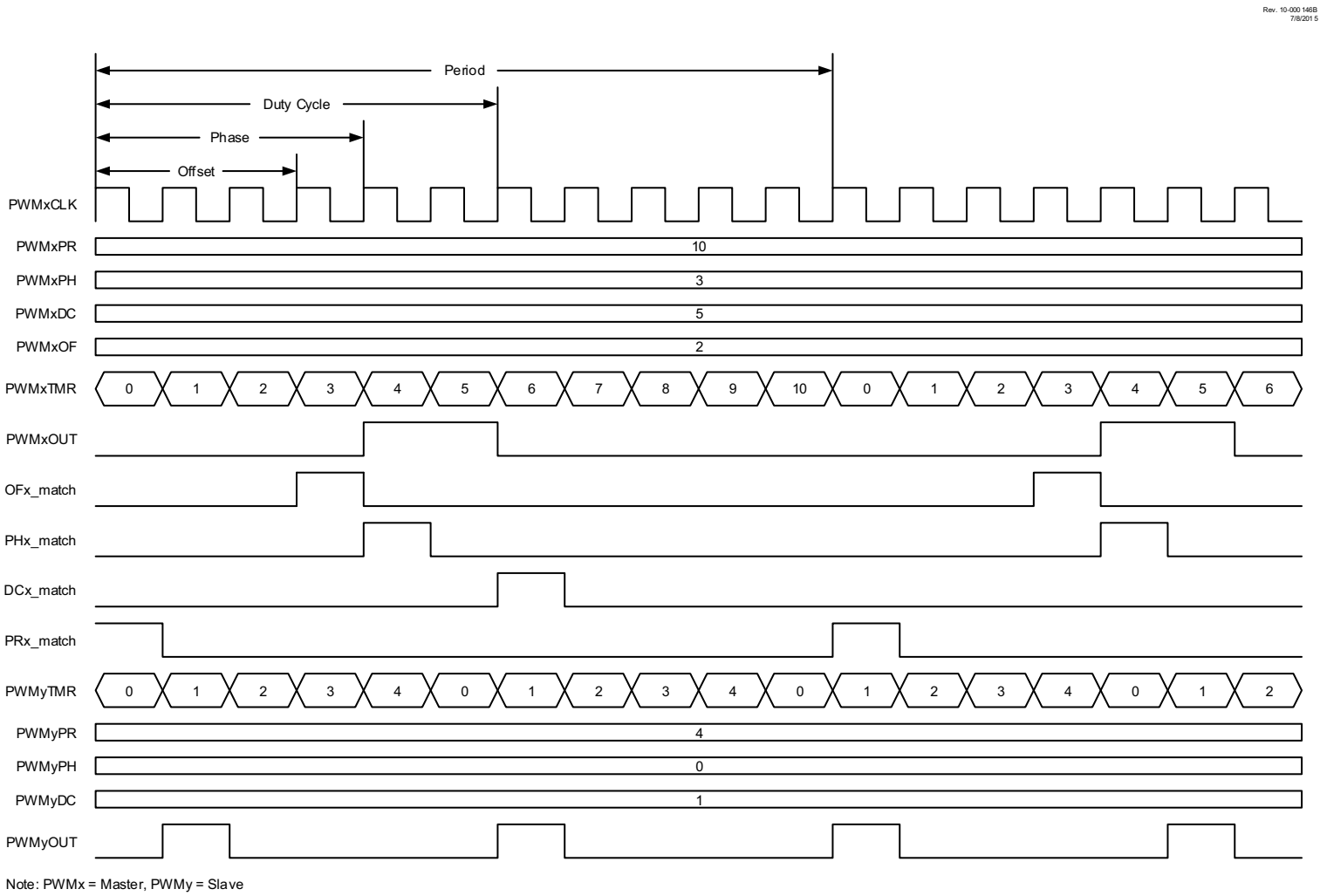
| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|---|---|---|---|---|---|---|---|---|---|
| BAUDCON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 186 |
| INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 74 |
| PIE1 | TMR1GIE | ADIE | RCIE(1) | TXIE(1) | — | — | TMR2IE | TMR1IE | 75 |
| PIR1 | TMR1GIF | ADIF | RCIF(1) | TXIF(1) | — | — | TMR2IF | TMR1IF | 78 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 185 |
| TXREG | EUSART Transmit Data Register | | | | | | | | 177* |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 184 |

**Legend:** — = unimplemented location, read as '0'. Shaded cells are not used for synchronous slave transmission.

* Page provides register information.

**Note 1:** PIC12(L)F1572 only.

**FIGURE 22-8: INDEPENDENT RUN MODE TIMING DIAGRAM**




Note: PWMx = Master, PWMy = Slave

**REGISTER 22-4: PWMxCLKCON: PWMx CLOCK CONTROL REGISTER**

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

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

bit 7 **Unimplemented:** Read as ‘0’

bit 6-4 **PS<2:0>:** Clock Source Prescaler Select bits
- 111 = Divides clock source by 128
- 110 = Divides clock source by 64
- 101 = Divides clock source by 32
- 100 = Divides clock source by 16
- 011 = Divides clock source by 8
- 010 = Divides clock source by 4
- 001 = Divides clock source by 2
- 000 = No prescaler

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

bit 1-0 **CS<1:0>:** Clock Source Select bits
- 11 = Reserved
- 10 = LFINTOSC (continues to operate during Sleep)
- 01 = HFINTOSC (continues to operate during Sleep)
- 00 = FOSC

**CALL** **Call Subroutine**

| | |
|---|---|
| Syntax: | [ *label* ] CALL k |
| Operands: | $0 \le k \le 2047$ |
| Operation: | (PC)+ 1→ TOS,<br>k → PC<10:0>,<br>(PCLATH<6:3>) → PC<14:11> |
| Status Affected: | None |
| Description: | Call Subroutine. First, return address (PC + 1) is pushed onto the stack. The 11-bit immediate address is loaded into PC bits <10:0>. The upper bits of the PC are loaded from PCLATH. CALL is a 2-cycle instruction. |

**CALLW** **Subroutine Call With W**

| | |
|---|---|
| Syntax: | [ *label* ] CALLW |
| Operands: | None |
| Operation: | (PC) +1 → TOS,<br>(W) → PC<7:0>,<br>(PCLATH<6:0>) → PC<14:8> |
| Status Affected: | None |
| Description: | Subroutine call with W. First, the return address (PC + 1) is pushed onto the return stack. Then, the contents of W is loaded into PC<7:0>, and the contents of PCLATH into PC<14:8>. CALLW is a 2-cycle instruction. |

**CLRF** **Clear f**

| | |
|---|---|
| Syntax: | [ *label* ] CLRF f |
| Operands: | $0 \le f \le 127$ |
| Operation: | 00h → (f)<br>1 → Z |
| Status Affected: | Z |
| Description: | The contents of register ‘f’ are cleared and the Z bit is set. |

**CLRW** **Clear W**

| | |
|---|---|
| Syntax: | [ *label* ] CLRW |
| Operands: | None |
| Operation: | 00h → (W)<br>1 → Z |
| Status Affected: | Z |
| Description: | W register is cleared. Zero bit (Z) is set. |

**CLRWDT** **Clear Watchdog Timer**

| | |
|---|---|
| Syntax: | [ *label* ] CLRWDT |
| Operands: | None |
| Operation: | 00h → WDT<br>0 → WDT prescaler,<br>1 → $\overline{TO}$<br>1 → $\overline{PD}$ |
| Status Affected: | $\overline{TO}$, $\overline{PD}$ |
| Description: | CLRWDT instruction resets the Watchdog Timer. It also resets the prescaler of the WDT.<br>Status bits $\overline{TO}$ and $\overline{PD}$ are set. |

**COMF** **Complement f**

| | |
|---|---|
| Syntax: | [ *label* ] COMF f,d |
| Operands: | $0 \le f \le 127$<br>$d \in [0,1]$ |
| Operation: | $(\bar{f}) \rightarrow$ (destination) |
| Status Affected: | Z |
| Description: | The contents of register ‘f’ are complemented. If ‘d’ is ‘0’, the result is stored in W. If ‘d’ is ‘1’, the result is stored back in register ‘f’. |

**DECF** **Decrement f**

| | |
|---|---|
| Syntax: | [ *label* ] DECF f,d |
| Operands: | $0 \le f \le 127$<br>$d \in [0,1]$ |
| Operation: | (f) - 1 → (destination) |
| Status Affected: | Z |
| Description: | Decrement register ‘f’. If ‘d’ is ‘0’, the result is stored in the W register. If ‘d’ is ‘1’, the result is stored back in register ‘f’. |

**MOVWI** **Move W to INDFn**

Syntax: [ *label* ] MOVWI ++FSRn
[ *label* ] MOVWI --FSRn
[ *label* ] MOVWI FSRn++
[ *label* ] MOVWI FSRn--
[ *label* ] MOVWI k[FSRn]

Operands: $n \in [0,1]$
$mm \in [00,01, 10, 11]$
$-32 \leq k \leq 31$

Operation: $W \rightarrow INDFn$
Effective address is determined by
- FSR + 1 (preincrement)
- FSR - 1 (predecrement)
- FSR + k (relative offset)

After the Move, the FSR value will be either:
- FSR + 1 (all increments)
- FSR - 1 (all decrements)

Unchanged

Status Affected: None

| Mode | Syntax | mm |
|---|---|---|
| Preincrement | ++FSRn | 00 |
| Predecrement | --FSRn | 01 |
| Postincrement | FSRn++ | 10 |
| Postdecrement | FSRn-- | 11 |

Description: This instruction is used to move data between W and one of the indirect registers (INDFn). Before/after this move, the pointer (FSRn) is updated by pre/post incrementing/decrementing it.

**Note:** The INDFn registers are not physical registers. Any instruction that accesses an INDFn register actually accesses the register at the address specified by the FSRn.

FSRn is limited to the range 0000h - FFFFh. Incrementing/decrementing it beyond these bounds will cause it to wraparound.

The increment/decrement operation on FSRn WILL NOT affect any Status bits.

**NOP** **No Operation**

Syntax: [ *label* ] NOP

Operands: None

Operation: No operation

Status Affected: None

Description: No operation.

Words: 1

Cycles: 1

Example:
```
NOP
```

**OPTION** **Load OPTION_REG Register with W**

Syntax: [ *label* ] OPTION

Operands: None

Operation: $(W) \rightarrow OPTION\_REG$

Status Affected: None

Description: Move data from W register to OPTION_REG register.

**RESET** **Software Reset**

Syntax: [ *label* ] RESET

Operands: None

Operation: Execute a device Reset. Resets the nRI flag of the PCON register.

Status Affected: None

Description: This instruction provides a way to execute a hardware Reset by software.

**TABLE 26-2: SUPPLY CURRENT ($I_{DD}$)(1,2) (CONTINUED)**

| PIC12LF1571/2 | | Standard Operating Conditions (unless otherwise stated) | | | | | |
|---|---|---|---|---|---|---|---|
| PIC12F1571/2 | | | | | | | |
| Param. No. | Device Characteristics | Min. | Typ† | Max. | Units | Conditions | |
| | | | | | | $V_{DD}$ | Note |
| D018A* | | — | 2 | 2.4 | mA | 3.0 | $F_{OSC}$ = 32 MHz, HFINTOSC **(Note 3)** |
| D018A* | | — | 2.1 | 2.5 | mA | 3.0 | $F_{OSC}$ = 32 MHz, HFINTOSC **(Note 3)** |
| | | — | 2.2 | 2.6 | mA | 5.0 | |
| D019A | | — | 1.7 | 1.9 | mA | 3.0 | $F_{OSC}$ = 32 MHz, External Clock (ECH), High-Power mode **(Note 3)** |
| D019A | | — | 1.8 | 2 | mA | 3.0 | $F_{OSC}$ = 32 MHz, External Clock (ECH), High-Power mode **(Note 3)** |
| | | — | 1.9 | 2.3 | mA | 5.0 | |
| D019B | | — | 2.2 | 5.9 | µA | 1.8 | $F_{OSC}$ = 32 kHz, External Clock (ECL), Low-Power mode |
| | | — | 4.3 | 8.3 | µA | 3.0 | |
| D019B | | — | 12 | 20 | µA | 2.3 | $F_{OSC}$ = 32 kHz, External Clock (ECL), Low-Power mode |
| | | — | 15 | 25 | µA | 3.0 | |
| | | — | 17 | 26 | µA | 5.0 | |
| D019C | | — | 18 | 25 | µA | 1.8 | $F_{OSC}$ = 500 kHz, External Clock (ECL), Low-Power mode |
| | | — | 30 | 38 | µA | 3.0 | |
| D019C | | — | 29 | 40 | µA | 2.3 | $F_{OSC}$ = 500 kHz, External Clock (ECL), Low-Power mode |
| | | — | 37 | 51 | µA | 3.0 | |
| | | — | 42 | 53 | µA | 5.0 | |

* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, +25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** The test conditions for all $I_{DD}$ measurements in active operation mode are: CLKIN = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to $V_{SS}$; $\overline{MCLR}$ = $V_{DD}$; WDT disabled.

**2:** The supply current is mainly a function of the operating voltage and frequency. Other factors, such as I/O pin loading and switching rate, oscillator type, internal code execution pattern and temperature, also have an impact on the current consumption.

**3:** PLL required for 32 MHz operation.

**FIGURE 27-3: IDD, EC OSCILLATOR, LOW-POWER MODE, FOSC = 500 kHz, PIC12LF1571/2 ONLY**

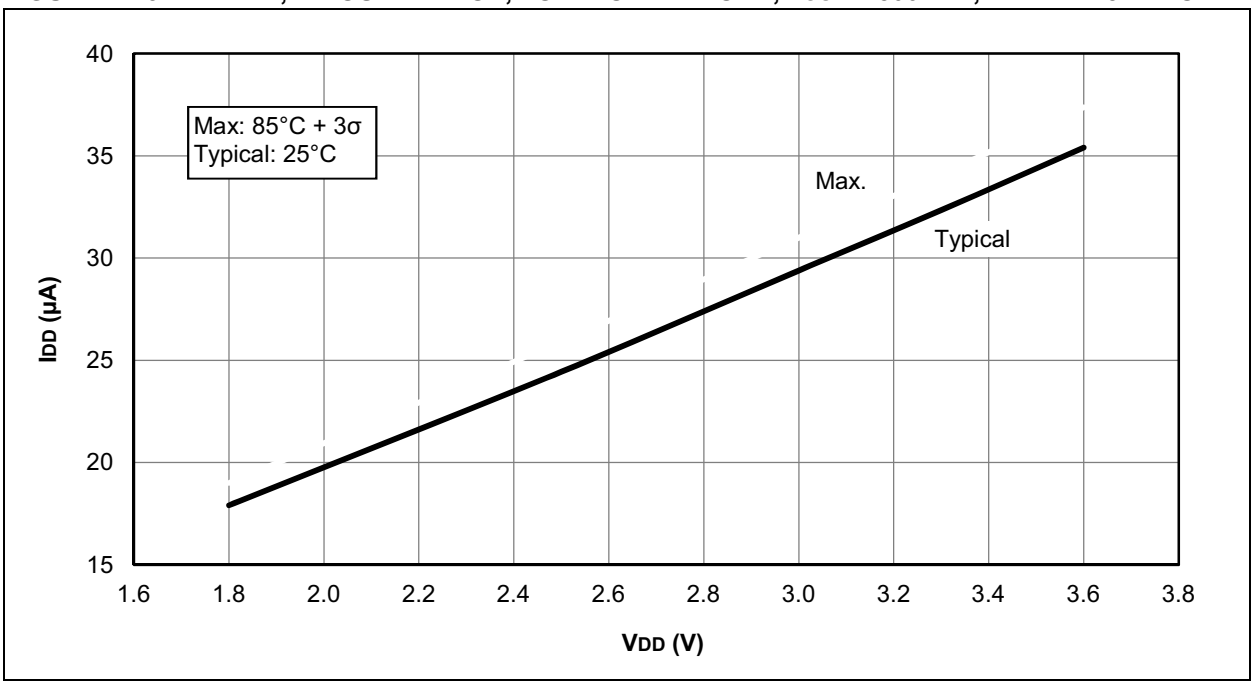



**FIGURE 27-4: IDD, EC OSCILLATOR, LOW-POWER MODE, FOSC = 500 kHz, PIC12F1571/2 ONLY**

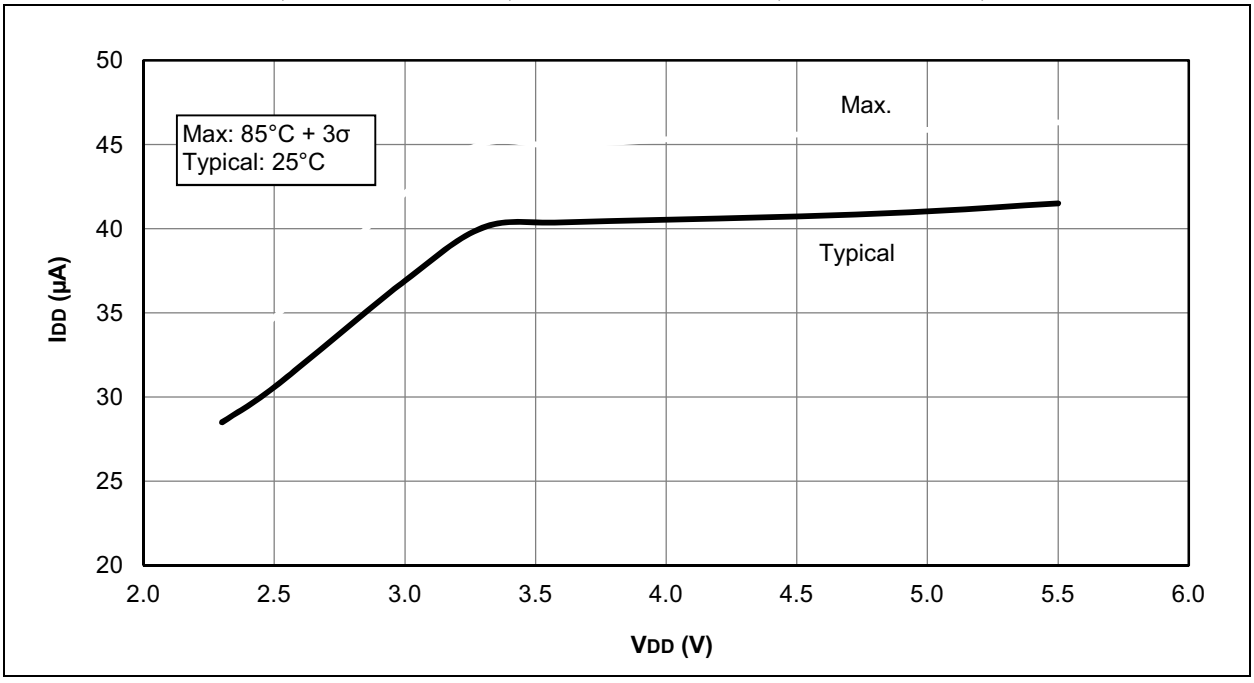


**FIGURE 27-7: IDD TYPICAL, EC OSCILLATOR, MEDIUM POWER MODE, PIC12F1571/2 ONLY**

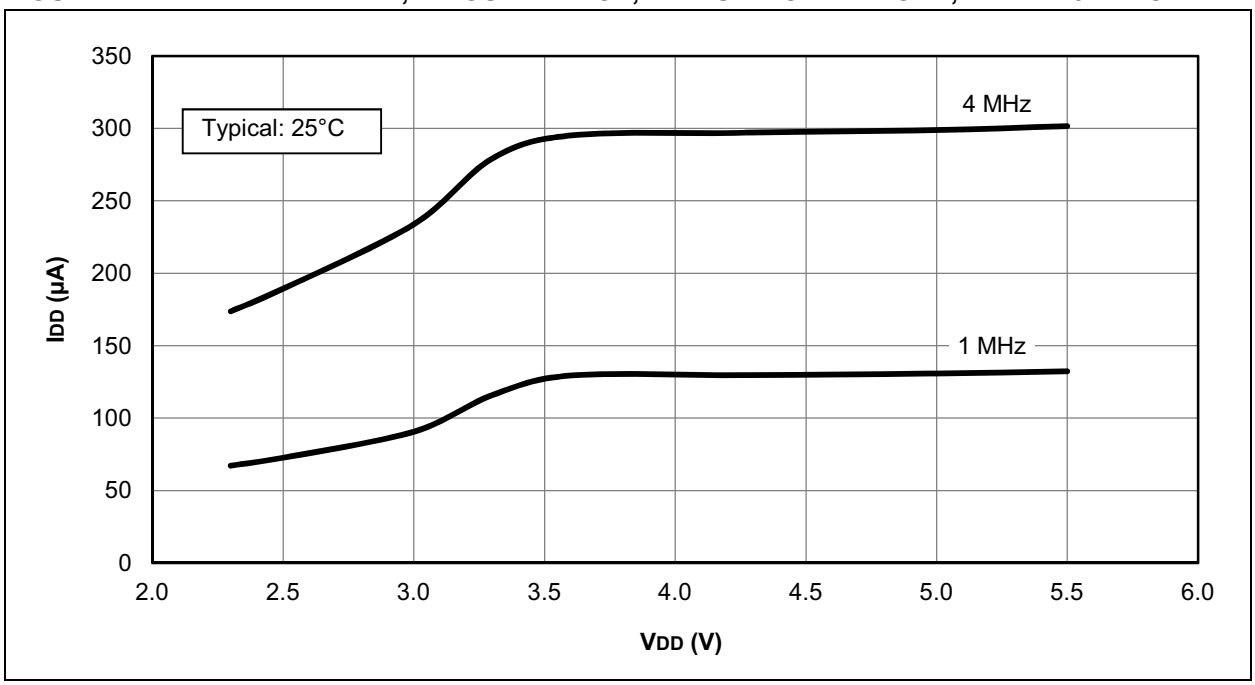
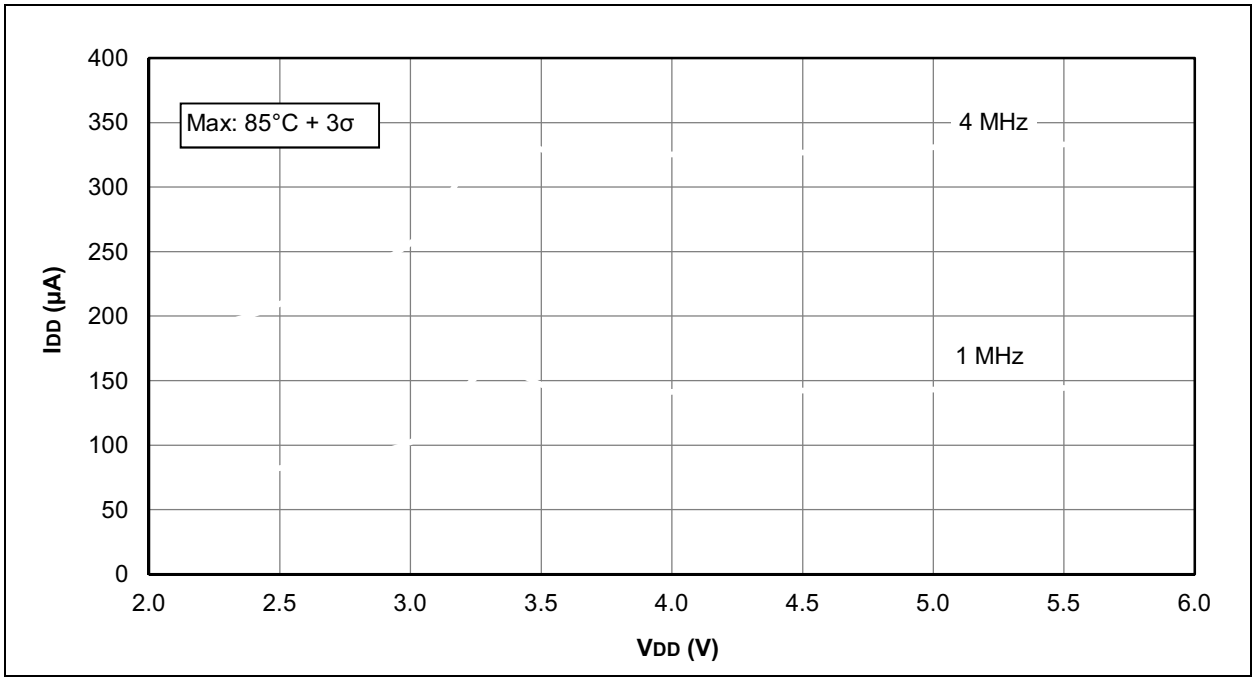



**FIGURE 27-8: IDD MAXIMUM, EC OSCILLATOR, MEDIUM POWER MODE, PIC12F1571/2 ONLY**



# APPENDIX A: DATA SHEET REVISION HISTORY

## Revision A (10/2013)

Original release of this document.

## Revision B (2/2014)

Updated PIC12(L)F1571/2 Family Types table Program Memory Flash heading (*words* to *K words*).

## Revision C (8/2014)

Updated PWM chapter. Changed to Final data sheet. Updated IDD and IPD parameters in the Electrical Specification chapter. Added Characterization Graphs.

Added Section 1.1: Register and Bit Naming Conventions.

Updated Figures 5-3 and 15-5. Updated Tables 3-1, 3-7, and 3-10. Updated Section 15.2.5. Updated Equation 15-1.

## Revision D (8/2015)

Updated Clocking Structure, Memory, Low-Power Features, Family Types table and Pin Diagram Table on cover pages.

Added Sections 3.2: High-Endurance Flash and 5.4: Clock Switching Before Sleep. Added Table 29-2 and 8-pin UDFN packaging.

Updated Examples 3-2 and 15-1.

Updated Figures 8-1, 21-1, 22-8 through 22-13 and 23-1.

Updated Registers 7-5, 8-1, 22-6 and 23-3.

Updated Sections 8.2.2, 15.2.6, 16.0, 21.0, 21.4.2, 22.3.3, 23.9.1.2, 23.11.1, 26.1 and 29.1.

Updated Tables 1, 3-3, 3-4, 3-10, 5-1, 16-1, 17-3, 22-2, 23-2, 26-6, 26-8 and 29-1.