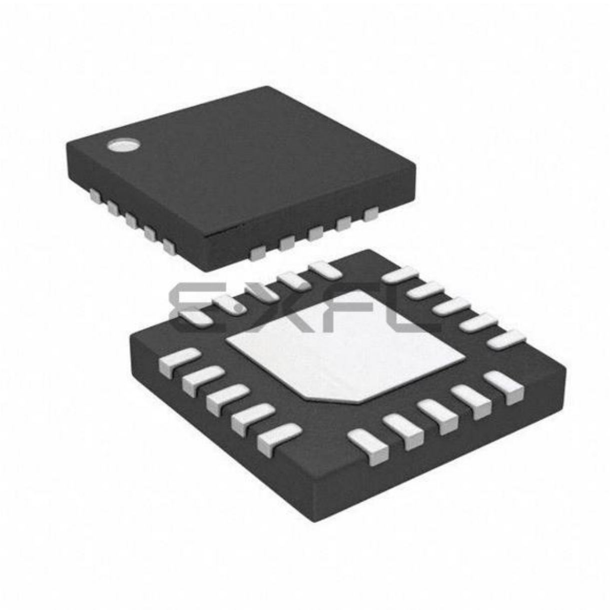




## What is "Embedded - Microcontrollers"?

"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 | I²C, LINbus, SPI, UART/USART |
| Peripherals | Brown-out Detect/Reset, POR, PWM, WDT |
| Number of I/O | 18 |
| Program Memory Size | 28KB (16K x 14) |
| Program Memory Type | FLASH |
| EEPROM Size | 256 x 8 |
| RAM Size | 2K x 8 |
| Voltage - Supply (Vcc/Vdd) | 1.8V ~ 3.6V |
| Data Converters | A/D 17x10b; D/A 1x5b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 20-UFQFN Exposed Pad |
| Supplier Device Package | 20-UQFN (4x4) |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/pic16lf18346-i-gz |

## TABLE 4-4: SPECIAL FUNCTION REGISTER SUMMARY BANKS 0-31 (CONTINUED)

| Address | Name | PIC16(L)F18326 | PIC16(L)F18346 | 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 7** | | | | | | | | | | | | | |
| | | | | | | **CPU CORE REGISTERS; see Table 4-2 for specifics** | | | | | | | |
| 38Ch | INLVLA | | | — | — | INLVLA5 | INLVLA4 | INLVLA3 | INLVLA2 | INLVLA1 | INLVLA0 | `--11 1111` | `--11 1111` |
| 38Dh | INLVLB | X | — | Unimplemented | | | | | | | | — | — |
| | | — | X | INLVLB7 | INLVLB6 | INLVLB5 | INLVLB4 | — | — | — | — | `1111 ----` | `1111 ----` |
| 38Eh | INLVLC | X | — | — | — | INLVLC5 | INLVLC4 | INLVLC3 | INLVLC2 | INLVLC1 | INLVLC0 | `--11 1111` | `--11 1111` |
| | | — | X | INLVLC7 | INLVLC6 | INLVLC5 | INLVLC4 | INLVLC3 | INLVLC2 | INLVLC1 | INLVLC0 | `1111 1111` | `1111 1111` |
| 38Fh | — | — | | Unimplemented | | | | | | | | — | — |
| 390h | — | — | | Unimplemented | | | | | | | | — | — |
| 391h | IOCAP | | | — | — | IOCAP5 | IOCAP4 | IOCAP3 | IOCAP2 | IOCAP1 | IOCAP0 | `--00 0000` | `--00 0000` |
| 392h | IOCAN | | | — | — | IOCAN5 | IOCAN4 | IOCAN3 | IOCAN2 | IOCAN1 | IOCAN0 | `--00 0000` | `--00 0000` |
| 393h | IOCAF | | | — | — | IOCAF5 | IOCAF4 | IOCAF3 | IOCAF2 | IOCAF1 | IOCAF0 | `--00 0000` | `--00 0000` |
| 394h | IOCBP | X | — | Unimplemented | | | | | | | | — | — |
| | | — | X | IOCBP7 | IOCBP6 | IOCBP5 | IOCBP4 | — | — | — | — | `0000 ----` | `0000 ----` |
| 395h | IOCBN | X | — | Unimplemented | | | | | | | | — | — |
| | | — | X | IOCBN7 | IOCBN6 | IOCBN5 | IOCBN4 | — | — | — | — | `0000 ----` | `0000 ----` |
| 396h | IOCBF | X | — | Unimplemented | | | | | | | | — | — |
| | | — | X | IOCBF7 | IOCBF6 | IOCBF5 | IOCBF4 | — | — | — | — | `0000 ----` | `0000 ----` |
| 397h | IOCCP | X | — | — | — | IOCCP5 | IOCCP4 | IOCCP3 | IOCCP2 | IOCCP1 | IOCCP0 | `--00 0000` | `--00 0000` |
| | | — | X | IOCCP7 | IOCCP6 | IOCCP5 | IOCCP4 | IOCCP3 | IOCCP2 | IOCCP1 | IOCCP0 | `0000 0000` | `0000 0000` |
| 398h | IOCCN | X | — | — | — | IOCCN5 | IOCCN4 | IOCCN3 | IOCCN2 | IOCCN1 | IOCCN0 | `--00 0000` | `--00 0000` |
| | | — | X | IOCCN7 | IOCCN6 | IOCCN5 | IOCCN4 | IOCCN3 | IOCCN2 | IOCCN1 | IOCCN0 | `0000 0000` | `0000 0000` |
| 399h | IOCCF | X | — | — | — | IOCCF5 | IOCCF4 | IOCCF3 | IOCCF2 | IOCCF1 | IOCCF0 | `--00 0000` | `--00 0000` |
| | | — | X | IOCCF7 | IOCCF6 | IOCCF5 | IOCCF4 | IOCCF3 | IOCCF2 | IOCCF1 | IOCCF0 | `0000 0000` | `0000 0000` |

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

**Note 1:** Only on PIC16F18326/18346.

**2:** Register accessible from both User and ICD Debugger.



Preliminary

### 7.2.2 INTERNAL CLOCK SOURCES

The device may be configured to use the internal oscillator block as the system clock by performing one of the following actions:

- Program the RSTOSC<2:0> bits in Configuration Words to select the INTOSC clock source, which will be used as the default system clock upon a device Reset.
- Write the NOSC<2:0> bits in the OSCCON1 register to switch the system clock source to the internal oscillator during run-time. See **Section 7.3 “Clock Switching”** for more information.

The function of the OSC2/CLKOUT pin is determined by the $\overline{\text{CLKOUTEN}}$ bit in Configuration Words.

The internal oscillator block has two independent oscillators that can produce two internal system clock sources.

1. The HFINTOSC (High-Frequency Internal Oscillator) is factory-calibrated and operates up to 32 MHz.
2. The LFINTOSC (Low-Frequency Internal Oscillator) is factory-calibrated and operates at 31 kHz.

#### 7.2.2.1 HFINTOSC

The High-Frequency Internal Oscillator (HFINTOSC) is a precision digitally-controlled internal clock source that produces a stable clock up to 32 MHz. The HFINTOSC can be enabled through one of the following methods:

- Programming the RSTOSC<2:0> bits in Configuration Word 1 to ‘110’ (1 MHz) or ‘000’ (32 MHz) to set the oscillator upon device Power-up or Reset
- Write to the NOSC<2:0> bits of the OSCCON1 register during run-time

The HFINTOSC frequency can be selected by setting the HFFRQ<3:0> bits of the OSCFRQ register.

The NDIV<3:0> bits of the OSCCON1 register allow for division of the output of the selected clock source by a range between 1:1 and 1:512.

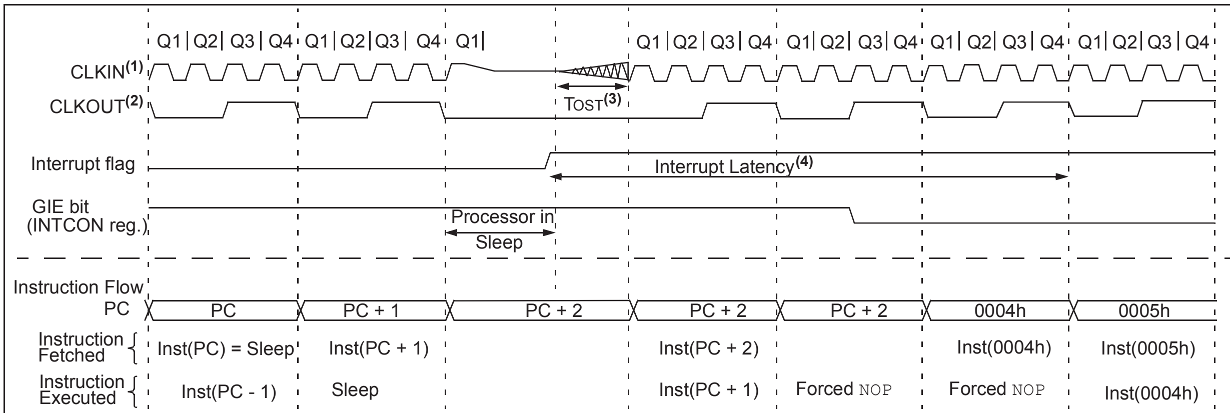
#### 7.2.2.2 2x PLL

The oscillator module contains a PLL that can be used with the HFINTOSC clock source to provide a system clock source. The input frequency to the PLL is limited to 8, 12, or 16 MHz, which will yield a system clock source of 16, 24, or 32 MHz, respectively.

The PLL may be enabled for use by one of two methods:

1. Program the RSTOSC bits in the Configuration Word 1 to ‘000’ to enable the HFINTOSC (32 MHz). This setting configures the HFFRQ<3:0> bits to ‘110’ (16 MHz) and activates the 2x PLL.
2. Write ‘000’ to the NOSC<2:0> bits in the OSCCON1 register to enable the 2x PLL, and write the correct value into the HFFRQ<3:0> bits of the OSCFRQ register to select the desired system clock frequency. See Register 6-6 for more information.

**FIGURE 9-2: WAKE-UP FROM SLEEP THROUGH INTERRUPT**




**Note 1:** External clock. High, Medium, Low mode assumed.
**2:** CLKOUT is shown here for timing reference.
**3:** $T_{OST}$ = 1024 $T_{OSC}$. This delay does not apply to EC and INTOSC Oscillator modes.
**4:** GIE = `1` assumed. In this case after wake-up, the processor calls the ISR at 0004h. If GIE = `0`, execution will continue in-line.

### 11.4.7 NVMREG EEPROM, USER ID, DEVICE ID AND CONFIGURATION WORD ACCESS

Instead of accessing Program Flash Memory, the EEPROM, the User ID's, Device ID/Revision ID and Configuration Words can be accessed when NVMREGS = 1 in the NVMCON1 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 11-3.

When read access is initiated on an address outside the parameters listed in Table 11-3, the NVMDATH: NVMDATL register pair is cleared, reading back '0's.

**TABLE 11-3: EEPROM, USER ID, DEV/REV ID AND CONFIGURATION WORD ACCESS (NVMREGS = 1)**

| Address | Function | Read Access | Write Access |
|---|---|---|---|
| 8000h-8003h | User IDs | Yes | Yes |
| 8005h-8006h | Device ID/Revision ID | Yes | No |
| 8007h-800Ah | Configuration Words 1-4 | Yes | No |
| F000h-F0FFh | EEPROM | Yes | Yes |

### REGISTER 12-13: WPUB: WEAK PULL-UP PORTB REGISTER

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| WPUB7 | WPUB6 | WPUB5 | WPUB4 | — | — | — | — |
| 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 **WPUB<7:4>**: Weak Pull-up Register bits
1 = Weak Pull-up enabled
0 = Weak Pull-up disabled

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

### REGISTER 12-14: ODCONB: PORTB OPEN-DRAIN CONTROL REGISTER

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| ODCB7 | ODCB6 | ODCB5 | ODCB4 | — | — | — | — |
| 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 **ODCB<7:4>:** PORTB Open-Drain Configuration bits
For RB<7:4> pins, respectively:
1 = Port pin operates as open-drain drive (sink current only)
0 = Port pin operates as standard push-pull drive (source and sink current)

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

### REGISTER 14-3: PMD2: PMD CONTROL REGISTER 2

| U-0 | R/W-0/0 | R/W-0/0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | U-0 |
|---|---|---|---|---|---|---|---|
| — | DACMD | ADCMD | — | — | CMP2MD | CMP1MD | — |
| 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 **Unimplemented:** Read as ‘0’

bit 6 **DACMD:** Disable DAC bit
1 = DAC module disabled
0 = DAC module enabled

bit 5 **ADCMD:** Disable ADC bit
1 = ADC module disabled
0 = ADC module enabled

bit 4-3 **Unimplemented:** Read as ‘0’

bit 2 **CMP2MD:** Disable Comparator C2 bit
1 = C2 module disabled
0 = C2 module enabled

bit 1 **CMP1MD:** Disable Comparator C1 bit
1 = C1 module disabled
0 = C1 module enabled

bit 0 **Unimplemented:** Read as ‘0’

**REGISTER 20-4: CWGxDAT: CWGx DATA INPUT SELECTION REGISTER**

| U-0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | DAT<3:0> | | | |
| bit 7 | | | | | | | bit 0 |

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

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

bit 3-0 **DAT<3:0>:** CWG Data Input Selection bits

| DAT | Data Source |
|---|---|
| 0000 | CWGxPPS |
| 0001 | C1OUT |
| 0010 | C2OUT |
| 0011 | CCP1 |
| 0100 | CCP2 |
| 0101 | CCP3 |
| 0110 | CCP4 |
| 0111 | PWM5 |
| 1000 | PWM6 |
| 1001 | NCO1 |
| 1010 | CLC1 |
| 1011 | CLC2 |
| 1100 | CLC3 |
| 1101 | CLC4 |
| 1110 | Reserved |
| 1111 | Reserved |

**TABLE 20-2: SUMMARY OF REGISTERS ASSOCIATED WITH CWGx**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|---|---|---|---|---|---|---|---|---|---|
| TRISA | — | — | TRISA5 | TRISA4 | —(2) | TRISA2 | TRISA1 | TRISA0 | 143 |
| ANSELA | — | — | ANSA5 | ANSA4 | — | ANSA2 | ANSA1 | ANSA0 | 144 |
| TRISB(1) | TRISB7 | TRISB6 | TRISB5 | TRISB4 | — | — | — | — | 149 |
| ANSELB(1) | ANSB7 | ANSB6 | ANSB5 | ANSB4 | — | — | — | — | 150 |
| TRISC | TRISC7(1) | TRISC6(1) | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 155 |
| ANSELC | ANSC7(1) | ANSC6(1) | ANSC5 | ANSC4 | ANSC3 | ANSC2 | ANSC1 | ANSC0 | 157 |
| PIR4 | CWG2IF | CWG1IF | TMR5GIF | TMR5IF | CCP4IF | CCP3IF | CCP2IF | CCP1IF | 110 |
| PIE4 | CWG2IE | CWG1IE | TMR5GIE | TMR5IE | CCP4IE | CCP3IE | CCP2IE | CCP1IE | 105 |
| CWG1CON0 | EN | LD | — | — | — | MODE<2:0> | | | 213 |
| CWG1CON1 | — | — | IN | — | POLD | POLC | POLB | POLA | 214 |
| CWG1CLKCON | — | — | — | — | — | — | — | CS | 214 |
| CWG1DAT | — | — | — | — | DAT<3:0> | | | | 215 |
| CWG1STR | OVRD | OVRC | OVRB | OVRA | STRD | STRC | STRB | STRA | 216 |
| CWG1AS0 | SHUTDOWN | REN | LSBD<1:0> | | LSAC<1:0> | | — | — | 217 |
| CWG1AS1 | — | — | — | AS4E | AS3E | AS2E | AS1E | AS0E | 218 |
| CWG1DBR | — | — | DBR<5:0> | | | | | | 218 |
| CWG1DBF | — | — | DBF<5:0> | | | | | | 219 |
| CWG1PPS | — | — | — | CWG1PPS<4:0> | | | | | 162 |
| CWG2CON0 | EN | LD | — | — | — | MODE<2:0> | | | 213 |
| CWG2CON1 | — | — | IN | — | POLD | POLC | POLB | POLA | 214 |
| CWG2CLKCON | — | — | — | — | — | — | — | CS | 214 |
| CWG2DAT | — | — | — | — | DAT<3:0> | | | | 215 |
| CWG2STR | OVRD | OVRC | OVRB | OVRA | STRD | STRC | STRB | STRA | 216 |
| CWG2AS0 | SHUTDOWN | REN | LSBD<1:0> | | LSAC<1:0> | | — | — | 217 |
| CWG2AS1 | — | — | — | AS4E | AS3E | AS2E | AS1E | AS0E | 218 |
| CWG2DBR | — | — | DBR<5:0> | | | | | | 218 |
| CWG2DBF | — | — | DBF<5:0> | | | | | | 219 |
| CWG2PPS | — | — | — | CWG2PPS<4:0> | | | | | 162 |

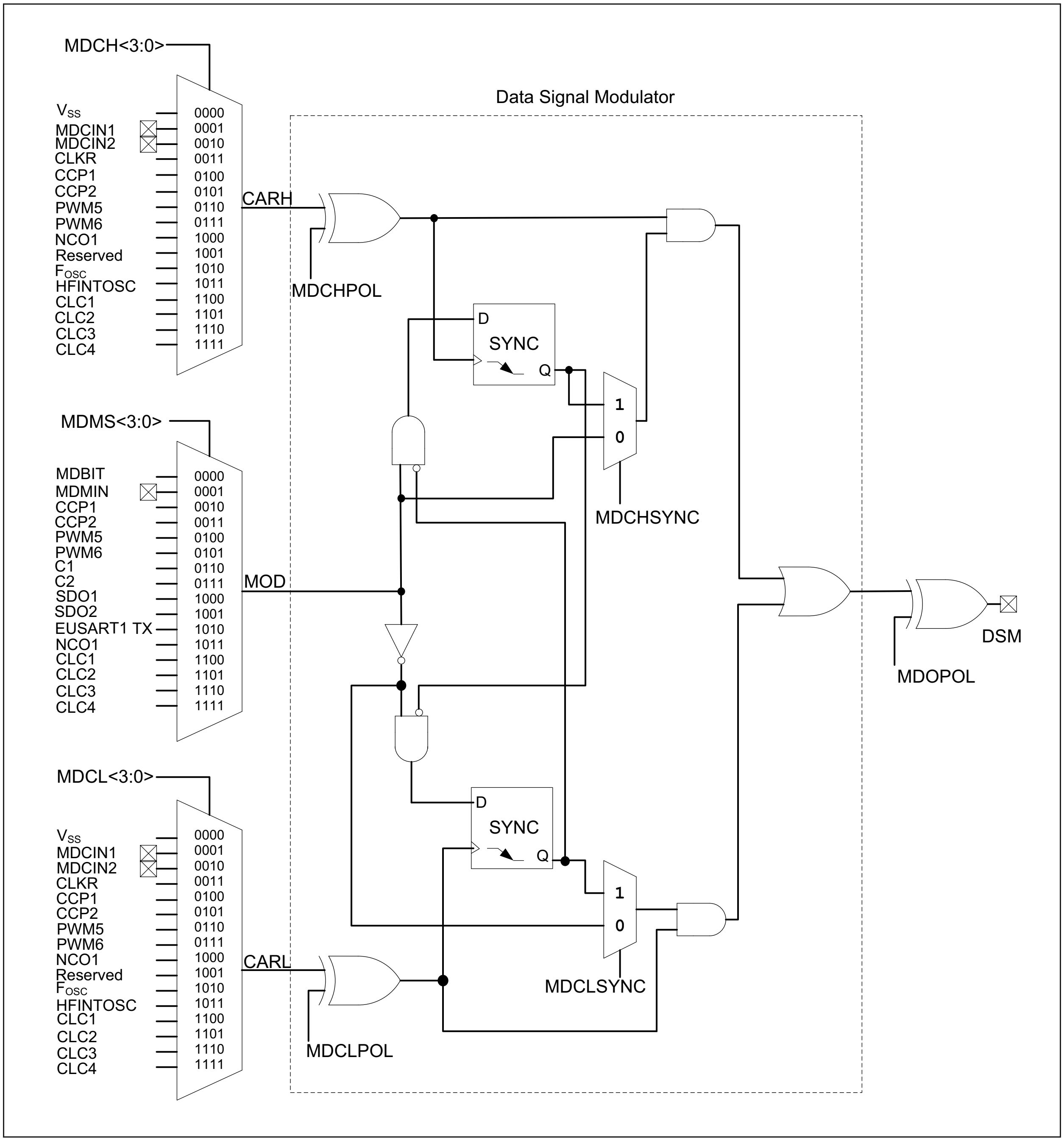
**Note 1:** PIC16(L)F18346 only.
**2:** Unimplemented, read as '0'.

**TABLE 22-3: SUMMARY OF REGISTERS ASSOCIATED WITH ADC**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE | PEIE | — | — | — | — | — | INTEDG | 100 |
| PIE1 | TMR1GIE | ADIE | RCIE | TXIE | SSP1IE | BCL1IE | TMR2IE | TMR1IE | 102 |
| PIR1 | TMR1GIF | ADIF | RCIF | TXIF | SSP1IF | BCL1IF | TMR2IF | TMR1IF | 107 |
| TRISA | — | — | TRISA5 | TRISA4 | —(2) | TRISA2 | TRISA1 | TRISA0 | 143 |
| TRISB(1) | TRISB7 | TRISB6 | TRISB5 | TRISB4 | — | — | — | — | 149 |
| TRISC | TRISC7(1) | TRISC6(1) | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 155 |
| ANSELA | — | — | ANSA5 | ANSA4 | — | ANSA2 | ANSA1 | ANSA0 | 144 |
| ANSELB(1) | ANSB7 | ANSB6 | ANSB5 | ANSB4 | — | — | — | — | 150 |
| ANSELC | ANSC7(1) | ANSC6(1) | ANSC5 | ANSC4 | ANSC3 | ANSC2 | ANSC1 | ANSC0 | 157 |
| ADCON0 | CHS<5:0> | | | | | | GO/$\overline{DONE}$ | ADON | 244 |
| ADCON1 | ADFM | ADCS<2:0> | | | — | ADNREF | ADPREF<1:0> | | 245 |
| ADACT | — | — | — | ADACT<4:0> | | | | | 246 |
| ADRESH | ADRESH<7:0> | | | | | | | | 247 |
| ADRESL | ADRESL<7:0> | | | | | | | | 247 |
| FVRCON | FVREN | FVRRDY | TSEN | TSRNG | CDAFVR<1:0> | | ADFVR<1:0> | | 180 |
| DAC1CON1 | — | — | — | DAC1R<4:0> | | | | | 264 |
| OSCSTAT1 | EXTOR | HFOR | — | LFOR | SOR | ADOR | — | PLLR | 91 |

**Legend:** – = unimplemented read as '0'. Shaded cells are not used for the ADC module.
**Note 1:** PIC16(L)F18346 only.
**2:** Unimplemented, read as '1'.

**FIGURE 25-1: SIMPLIFIED BLOCK DIAGRAM OF THE DATA SIGNAL MODULATOR**



## REGISTER 26-4: T0CON1: TIMER0 CONTROL REGISTER 1

| 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 |
|---|---|---|---|---|---|---|---|
| T0CS<2:0> | | | T0ASYNC | T0CKPS<3:0> | | | |
| bit 7 | | | | | | | bit 0 |

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

bit 7-5 **T0CS<2:0>:** Timer0 Clock Source Select bits
111 = CLC1
110 = SOSC
101 = Reserved
100 = LFINTOSC
011 = HFINTOSC
010 = FOSC/4
001 = T0CKIPPS (Inverted)
000 = T0CKIPPS (True)

bit 4 **T0ASYNC:** TMR0 Input Asynchronization Enable bit
1 = The input to the TMR0 counter is not synchronized to system clocks
0 = The input to the TMR0 counter is synchronized to FOSC/4

bit 3-0 **T0CKPS<3:0>:** Prescaler Rate Select bit
1111 = 1:32768
1110 = 1:16384
1101 = 1:8192
1100 = 1:4096
1011 = 1:2048
1010 = 1:1024
1001 = 1:512
1000 = 1:256
0111 = 1:128
0110 = 1:64
0101 = 1:32
0100 = 1:16
0011 = 1:8
0010 = 1:4
0001 = 1:2
0000 = 1:1

**FIGURE 27-4: TIMER1 GATE TOGGLE MODE**

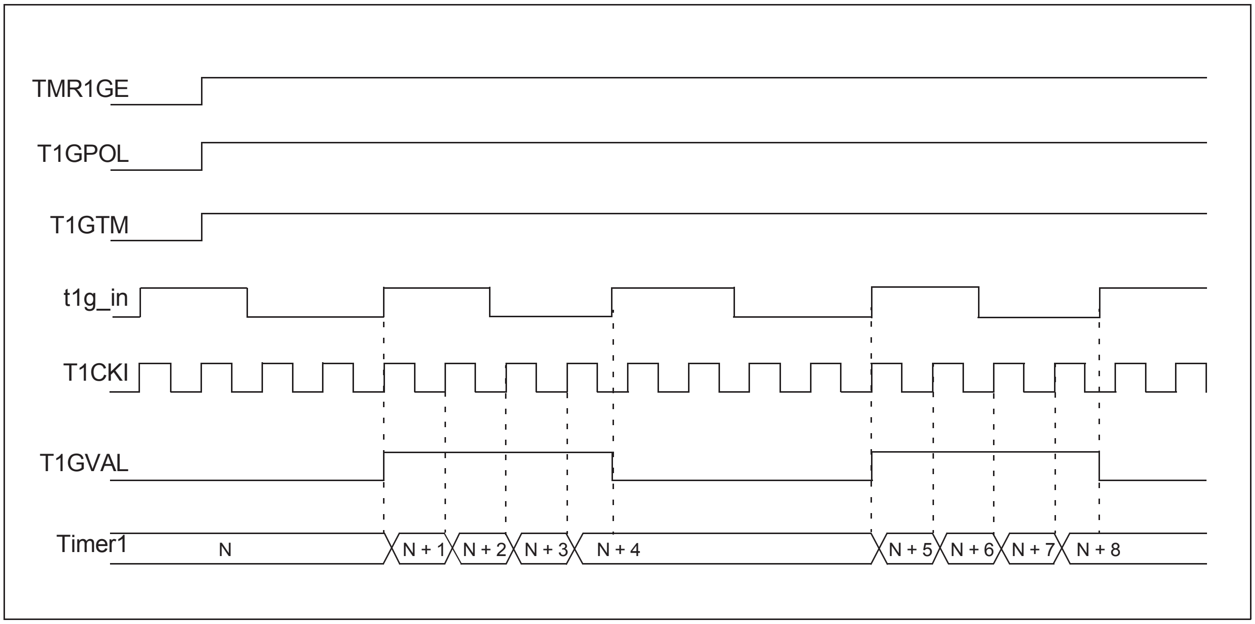
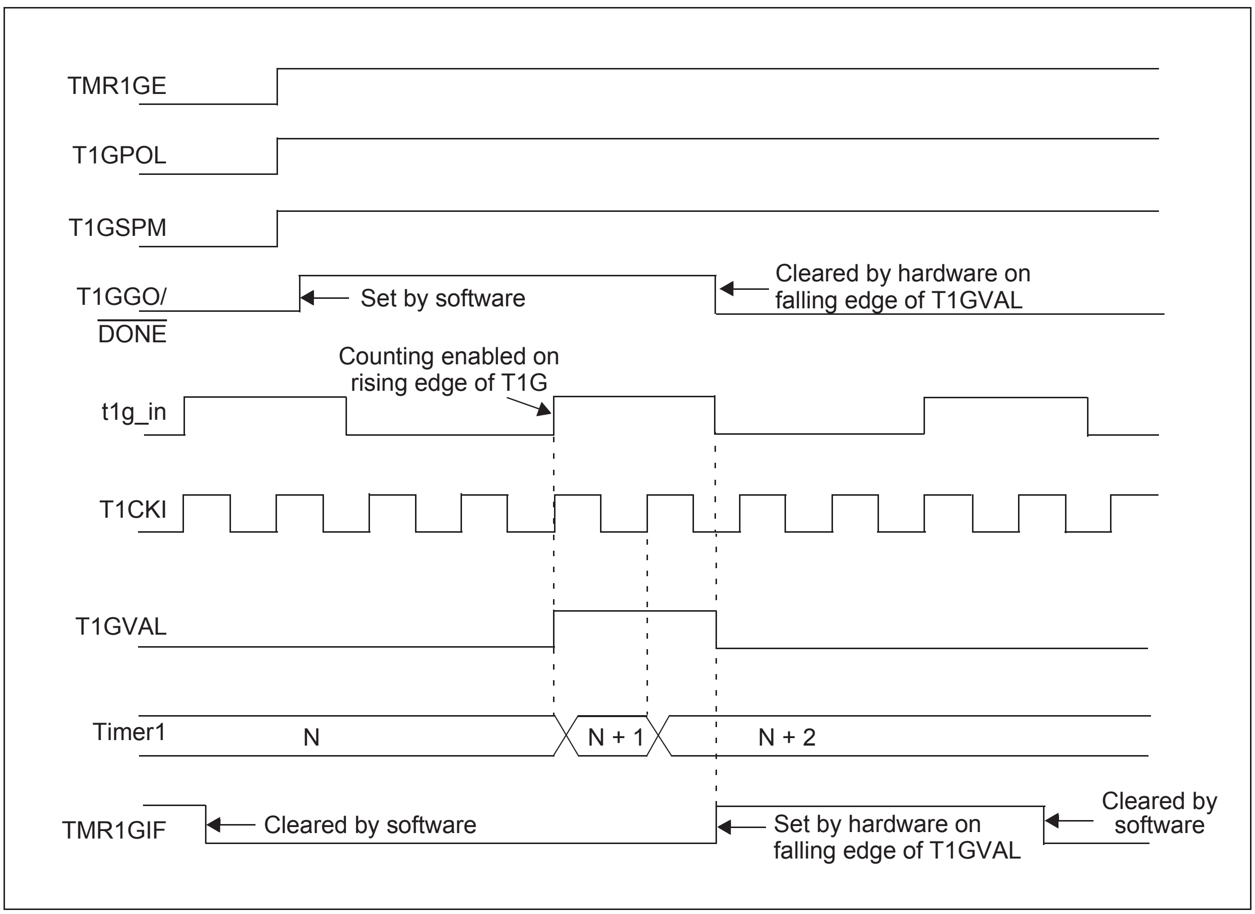



**FIGURE 27-5: TIMER1 GATE SINGLE-PULSE MODE**



## 28.5 Register Definitions: Timer2/4/6 Control

### REGISTER 28-1: TxCON(1): TIMERx CONTROL REGISTER

| U-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 |
|---|---|---|---|---|---|---|---|
| — | TxOUTPS<3:0> | | | | TMRxON | TxCKPS<1:0> | |
| bit 7 | | | | | | | bit 0 |

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

bit 7 **Unimplemented:** Read as ‘0’

bit 6-3 **TxOUTPS<3:0>:** Timerx Output Postscaler Select bits
1111 = 1:16 Postscaler
1110 = 1:15 Postscaler
1101 = 1:14 Postscaler
1100 = 1:13 Postscaler
1011 = 1:12 Postscaler
1010 = 1:11 Postscaler
1001 = 1:10 Postscaler
1000 = 1:9 Postscaler
0111 = 1:8 Postscaler
0110 = 1:7 Postscaler
0101 = 1:6 Postscaler
0100 = 1:5 Postscaler
0011 = 1:4 Postscaler
0010 = 1:3 Postscaler
0001 = 1:2 Postscaler
0000 = 1:1 Postscaler

bit 2 **TMRxON:** Timer2 On bit
1 = Timerx is ON
0 = Timerx is OFF

bit 1-0 **TxCKPS<1:0>:** Timerx Clock Prescale Select bits
11 = Prescaler is 64
10 = Prescaler is 16
01 = Prescaler is 4
00 = Prescaler is 1

**Note 1:** ‘x’ refers to either ‘2,’ 4’ or ‘6’ for the respective Timer2/4/6 registers.

## 29.5 Register Definitions: CCP Control

### REGISTER 29-1: CCPxCON: CCPx CONTROL REGISTER

| R/W-0/0 | U-0 | R-x/x | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| CCPxEN | — | CCPxOUT | CCPxFMT | CCPxMODE<3:0> | | | |
| bit 7 | | | | | | | bit 0 |

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

bit 7 **CCPxEN:** CCP Module Enable bit
1 = CCP is enabled
0 = CCP is disabled

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

bit 5 **CCPxOUT:** CCPx Output Data (read-only) bit

bit 4 **CCPxFMT:** CCPW (pulse width) Alignment bit
CCPxMODE = Capture mode
Unused
CCPxMODE = Compare mode
Unused
CCPxMODE = PWM mode
1 = Left-aligned format
0 = Right-aligned format

bit 3-0 **CCPxMODE<3:0>:** CCPx Mode Select bits(1)
1111 = PWM mode
1110 = Reserved
1101 = Reserved
1100 = Reserved

1011 = Compare mode: output will pulse 0-1-0; Clears TMR1/3/5
1010 = Compare mode: output will pulse 0-1-0
1001 = Compare mode: clear output on compare match
1000 = Compare mode: set output on compare match

0111 = Capture mode: every 16th rising edge of CCPx input
0110 = Capture mode: every 4th rising edge of CCPx input
0101 = Capture mode: every rising edge of CCPx input
0100 = Capture mode: every falling edge of CCPx input

0011 = Capture mode: every edge of CCPx input
0010 = Compare mode: toggle output on match
0001 = Compare mode: toggle output on match; clear TMR1/3/5
0000 = Capture/Compare/PWM off (resets CCPx module)

**Note 1:** All modes will set the CCPxIF bit and will trigger an ADC conversion if CCPx is selected as the ADC trigger source.

**REGISTER 31-2: RC1STA: RECEIVE STATUS AND CONTROL REGISTER**

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R-0/0 | R-0/0 | R-x/x |
|---|---|---|---|---|---|---|---|
| SPEN(1) | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D |
| 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 **SPEN:** Serial Port Enable bit(1)
1 = Serial port enabled
0 = Serial port disabled (held in Reset)

bit 6 **RX9:** 9-bit Receive Enable bit
1 = Selects 9-bit reception
0 = Selects 8-bit reception

bit 5 **SREN:** Single Receive Enable bit
Asynchronous mode:
Unused in this mode – value ignored
Synchronous mode – Master:
1 = Enables single receive
0 = Disables single receive
This bit is cleared after reception is complete.
Synchronous mode – Slave
Unused in this mode – value ignored

bit 4 **CREN:** Continuous Receive Enable bit
Asynchronous mode:
1 = Enables continuous receive until enable bit CREN is cleared
0 = Disables continuous receive
Synchronous mode:
1 = Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN)
0 = Disables continuous receive

bit 3 **ADDEN:** Address Detect Enable bit
Asynchronous mode 9-bit (RX9 = 1):
1 = Enables address detection – enable interrupt and load of the receive buffer when the ninth bit in the receive buffer is set
0 = Disables address detection, all bytes are received and ninth bit can be used as parity bit
Asynchronous mode 8-bit (RX9 = 0):
Unused in this mode – value ignored

bit 2 **FERR:** Framing Error bit
1 = Framing error (can be updated by reading RC1REG register and receive next valid byte)
0 = No framing error

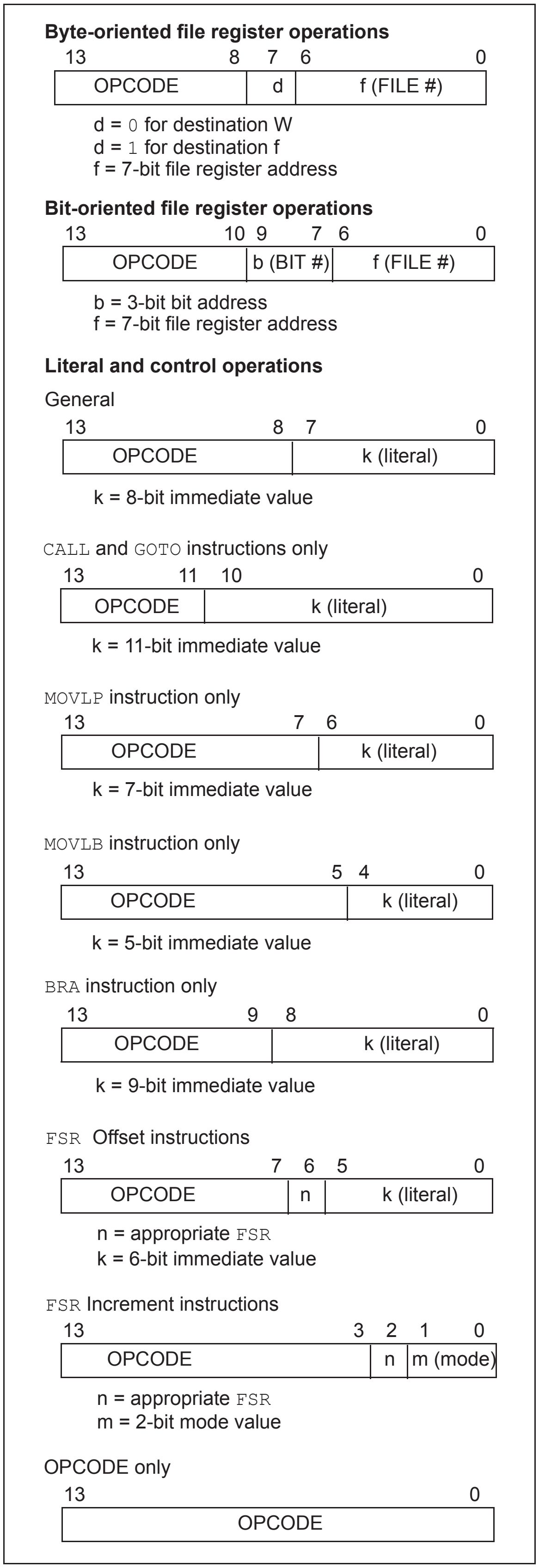
bit 1 **OERR:** Overrun Error bit
1 = Overrun error (can be cleared by clearing bit CREN)
0 = No overrun error

bit 0 **RX9D:** Ninth bit of Received Data
This can be address/data bit or a parity bit and must be calculated by user firmware.

**Note 1:** The EUSART1 module automatically changes the pin from tri-state to drive as needed. Configure the associated TRIS bits for TX/CK and RX/DT to 1.

FIGURE 34-1: GENERAL FORMAT FOR INSTRUCTIONS



### BCF — Bit Clear f

| | |
|---|---|
| Syntax: | [ *label* ] BCF f,b |
| Operands: | 0 ≤ f ≤ 127<br>0 ≤ b ≤ 7 |
| Operation: | `0` → (f<b>) |
| Status Affected: | None |
| Description: | Bit ‘b’ in register ‘f’ is cleared. |

### BRA — Relative Branch

| | |
|---|---|
| Syntax: | [ *label* ] BRA label<br>[ *label* ] BRA $+k |
| Operands: | -256 ≤ label - PC + 1 ≤ 255<br>-256 ≤ k ≤ 255 |
| Operation: | (PC) + 1 + k → PC |
| Status Affected: | None |
| Description: | Add the signed 9-bit literal ‘k’ to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 1 + k. This instruction is a 2-cycle instruction. This branch has a limited range. |

### BRW — Relative Branch with W

| | |
|---|---|
| Syntax: | [ *label* ] BRW |
| Operands: | None |
| Operation: | (PC) + (W) → PC |
| Status Affected: | None |
| Description: | Add the contents of W (unsigned) to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 1 + (W). This instruction is a 2-cycle instruction. |

### BSF — Bit Set f

| | |
|---|---|
| Syntax: | [ *label* ] BSF f,b |
| Operands: | 0 ≤ f ≤ 127<br>0 ≤ b ≤ 7 |
| Operation: | `1` → (f<b>) |
| Status Affected: | None |
| Description: | Bit ‘b’ in register ‘f’ is set. |

### BTFSC — Bit Test f, Skip if Clear

| | |
|---|---|
| Syntax: | [ *label* ] BTFSC f,b |
| Operands: | 0 ≤ f ≤ 127<br>0 ≤ b ≤ 7 |
| Operation: | skip if (f<b>) = `0` |
| Status Affected: | None |
| Description: | If bit ‘b’ in register ‘f’ is ‘`1`’, the next instruction is executed.<br>If bit ‘b’, in register ‘f’, is ‘`0`’, the next instruction is discarded, and a `NOP` is executed instead, making this a 2-cycle instruction. |

### BTFSS — Bit Test f, Skip if Set

| | |
|---|---|
| Syntax: | [ *label* ] BTFSS f,b |
| Operands: | 0 ≤ f ≤ 127<br>0 ≤ b < 7 |
| Operation: | skip if (f<b>) = `1` |
| Status Affected: | None |
| Description: | If bit ‘b’ in register ‘f’ is ‘`0`’, the next instruction is executed.<br>If bit ‘b’ is ‘`1`’, then the next instruction is discarded and a `NOP` is executed instead, making this a 2-cycle instruction. |

### CALL — Call Subroutine

| | |
|---|---|
| Syntax: | [ *label* ] CALL k |
| Operands: | 0 ≤ k ≤ 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 ≤ f ≤ 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. Status bits $\overline{TO}$ and $\overline{PD}$ are set. |

### COMF — Complement f

| | |
|---|---|
| Syntax: | [ *label* ] COMF f,d |
| Operands: | 0 ≤ f ≤ 127<br>d ∈ [`0`,`1`] |
| Operation: | $(\overline{f})$ → (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 ≤ f ≤ 127<br>d ∈ [`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’. |

**TABLE 35-12: ANALOG-TO-DIGITAL CONVERTER (ADC) CHARACTERISTICS(1,2)**

| Standard Operating Conditions (unless otherwise stated)<br>VDD = 3.0V, TA = 25°C | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param. No.** | **Sym.** | **Characteristic** | **Min.** | **Typ.†** | **Max.** | **Units** | **Conditions** |
| AD01 | NR | Resolution | — | — | 10 | bit | |
| AD02 | EIL | Integral Error | — | ±0.1 | ±1.0 | LSb | ADCREF+ = 3.0V, ADCREF- = 0V |
| AD03 | EDL | Differential Error | — | ±0.1 | ±1.0 | LSb | ADCREF+ = 3.0V, ADCREF-= 0V |
| AD04 | EOFF | Offset Error | — | 0.5 | 2 | LSb | ADCREF+ = 3.0V, ADCREF- = 0V |
| AD05 | EGN | Gain Error | — | ±0.2 | ±1.0 | LSb | ADCREF+ = 3.0V, ADCREF- = 0V |
| AD06 | VADREF | ADC Reference Voltage (ADREF+)(3) | 1.8 | — | VDD | V | |
| AD07 | VAIN | Full-Scale Range | VSS | — | ADREF+ | V | |
| AD06 | VADREF | ADC Reference Voltage (ADREF+ - ADREF-)(3) | 1.8 | — | VDD | V | |
| AD07 | VAIN | Full-Scale Range | ADREF- | — | ADREF+ | V | |
| AD08 | ZAIN | Recommended Impedance of Analog Voltage Source | — | 10 | — | kΩ | |
| AD09 | RVREF | ADC Voltage Reference Ladder Impedance | — | — | — | kΩ | |

* These parameters are characterized but not tested.

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

**Note 1:** Total Absolute Error is the sum of the offset, gain and integral non-linearity (INL) errors.

**2:** The ADC conversion result never decreases with an increase in the input and has no missing codes.

## 37.6 MPLAB X SIM Software Simulator

The MPLAB X SIM Software Simulator allows code development in a PC-hosted environment by simulating the PIC MCUs and dsPIC DSCs on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a comprehensive stimulus controller. Registers can be logged to files for further run-time analysis. The trace buffer and logic analyzer display extend the power of the simulator to record and track program execution, actions on I/O, most peripherals and internal registers.

The MPLAB X SIM Software Simulator fully supports symbolic debugging using the MPLAB XC Compilers, and the MPASM and MPLAB Assemblers. The software simulator offers the flexibility to develop and debug code outside of the hardware laboratory environment, making it an excellent, economical software development tool.

## 37.7 MPLAB REAL ICE In-Circuit Emulator System

The MPLAB REAL ICE In-Circuit Emulator System is Microchip's next generation high-speed emulator for Microchip Flash DSC and MCU devices. It debugs and programs all 8, 16 and 32-bit MCU, and DSC devices with the easy-to-use, powerful graphical user interface of the MPLAB X IDE.

The emulator is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with either a connector compatible with in-circuit debugger systems (RJ-11) or with the new high-speed, noise tolerant, Low-Voltage Differential Signal (LVDS) interconnection (CAT5).

The emulator is field upgradeable through future firmware downloads in MPLAB X IDE. MPLAB REAL ICE offers significant advantages over competitive emulators including full-speed emulation, run-time variable watches, trace analysis, complex breakpoints, logic probes, a ruggedized probe interface and long (up to three meters) interconnection cables.

## 37.8 MPLAB ICD 3 In-Circuit Debugger System

The MPLAB ICD 3 In-Circuit Debugger System is Microchip's most cost-effective, high-speed hardware debugger/programmer for Microchip Flash DSC and MCU devices. It debugs and programs PIC Flash microcontrollers and dsPIC DSCs with the powerful, yet easy-to-use graphical user interface of the MPLAB IDE.

The MPLAB ICD 3 In-Circuit Debugger probe is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with a connector compatible with the MPLAB ICD 2 or MPLAB REAL ICE systems (RJ-11). MPLAB ICD 3 supports all MPLAB ICD 2 headers.

## 37.9 PICkit 3 In-Circuit Debugger/Programmer

The MPLAB PICkit 3 allows debugging and programming of PIC and dsPIC Flash microcontrollers at a most affordable price point using the powerful graphical user interface of the MPLAB IDE. The MPLAB PICkit 3 is connected to the design engineer's PC using a full-speed USB interface and can be connected to the target via a Microchip debug (RJ-11) connector (compatible with MPLAB ICD 3 and MPLAB REAL ICE). The connector uses two device I/O pins and the Reset line to implement in-circuit debugging and In-Circuit Serial Programming™ (ICSP™).

## 37.10 MPLAB PM3 Device Programmer

The MPLAB PM3 Device Programmer is a universal, CE compliant device programmer with programmable voltage verification at $V_{DDMIN}$ and $V_{DDMAX}$ for maximum reliability. It features a large LCD display (128 x 64) for menus and error messages, and a modular, detachable socket assembly to support various package types. The ICSP cable assembly is included as a standard item. In Stand-Alone mode, the MPLAB PM3 Device Programmer can read, verify and program PIC devices without a PC connection. It can also set code protection in this mode. The MPLAB PM3 connects to the host PC via an RS-232 or USB cable. The MPLAB PM3 has high-speed communications and optimized algorithms for quick programming of large memory devices, and incorporates an MMC card for file storage and data applications.