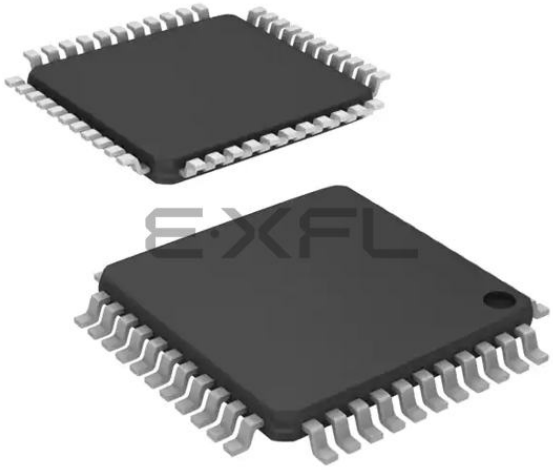


**Welcome to E-XFL.COM**

## 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 | I²C, LINbus, SPI, UART/USART |
| Peripherals | Brown-out Detect/Reset, POR, PWM, WDT |
| Number of I/O | 36 |
| Program Memory Size | 14KB (8K x 14) |
| Program Memory Type | FLASH |
| EEPROM Size | 224 x 8 |
| RAM Size | 1K x 8 |
| Voltage - Supply (Vcc/Vdd) | 1.8V ~ 3.6V |
| Data Converters | A/D 35x10b; D/A 1x5b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 125°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 44-TQFP |
| Supplier Device Package | 44-TQFP (10x10) |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/pic16lf15375-e-pt |

**TABLE 5: 48-PIN ALLOCATION TABLE (PIC16(L)F15385, PIC16(L)F15386) (CONTINUED)**

| I/O[2] | 48-Pin UQFN/TQFP | ADC | Reference | Comparator | NCO | DAC | Timers | CCP | PWM | CWG | MSSP | ZCD | EUSART | CLC | CLKR | Interrupt | Pull-up | Basic |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RF5 | 13 | ANF5 | — | — | — | — | — | — | — | — | — | — | — | — | — | — | Y | — |
| RF6 | 14 | ANF6 | — | — | — | — | — | — | — | — | — | — | — | — | — | — | Y | — |
| RF7 | 15 | ANF7 | — | — | — | — | — | — | — | — | — | — | — | — | — | — | Y | — |
| VDD | 30 | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | Y | VDD |
| VDD | 7 | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | VDD |
| VSS | 6 | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | VSS |
| VSS | 31 | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | VSS |
| OUT[2] | — | — | — | C1OUT | NCO1OUT | — | TMR0 | CCP1 | PWM3OUT | CWG1A<br>CWG2A | SDO1<br>SDO2 | — | DT[3] | CLC1OUT | CLKR | — | — | — |
| | — | — | — | C2OUT | — | — | — | CCP2 | PWM4OUT | CWG1B<br>CWG2B | SCK1<br>SCK2 | — | CK1<br>CK2 | CLC2OUT | — | — | — | — |
| | — | — | — | — | — | — | — | — | PWM5OUT | CWG1C<br>CWG2C | SCK1[3,4]<br>SCL2[3,4] | — | TX1<br>TX2 | CLC3OUT | — | — | — | — |
| | — | — | — | — | — | — | — | — | PWM6OUT | CWG1D<br>CWG2D | SDA1[3,4]<br>SDA2[3,4] | — | — | CLC4OUT | — | — | — | — |

**Note 1:** This is a PPS re-mappable input signal. The input function may be moved from the default location shown to one of several other PORTx pins.

**2:** All digital output signals shown in this row are PPS re-mappable. These signals may be mapped to output onto one of several PORTx pin options.

**3:** This is a bidirectional signal. For normal module operation, the firmware should map this signal to the same pin in both the PPS input and PPS output registers.

**4:** These pins are configured for $I^2C$ logic levels. PPS assignments to the other pins will operate, but input logic levels will be standard TTL/ST as selected by the INLVL register, instead of the $I^2C$ specific or SMBUS input buffer thresholds.

# 2.0 GUIDELINES FOR GETTING STARTED WITH PIC16(L)F15356/75/76/85/86 MICROCONTROLLERS

## 2.1 Basic Connection Requirements

Getting started with the PIC16(L)F15356/75/76/85/86 family of 8-bit microcontrollers requires attention to a minimal set of device pin connections before proceeding with development.

The following pins must always be connected:

- All VDD and VSS pins (see **Section 2.2 "Power Supply Pins"**)
- $\overline{\text{MCLR}}$ pin (see **Section 2.3 "Master Clear (MCLR) Pin"**)

These pins must also be connected if they are being used in the end application:

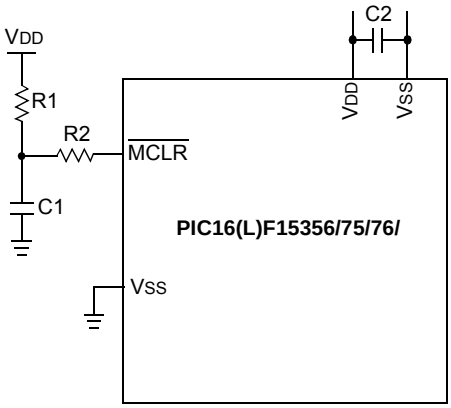
- PGC/PGD pins used for In-Circuit Serial Programming™ (ICSP™) and debugging purposes (see **Section 2.4 "ICSP™ Pins"**)
- OSCI and OSCO pins when an external oscillator source is used (see **Section 2.5 "External Oscillator Pins"**)

Additionally, the following pins may be required:

- VREF+/VREF- pins are used when external voltage reference for analog modules is implemented

The minimum mandatory connections are shown in Figure 2-1.

**FIGURE 2-1: RECOMMENDED MINIMUM CONNECTIONS**




**Key (all values are recommendations):**

C1: 10nF, 16V ceramic

C2: 0.1uF, 16V ceramic

R1: 10 kΩ

R2: 100Ω to 470Ω

## 2.2 Power Supply Pins

### 2.2.1 DECOUPLING CAPACITORS

The use of decoupling capacitors on every pair of power supply pins (VDD and VSS) is required.

Consider the following criteria when using decoupling capacitors:

- **Value and type of capacitor:** A 0.1 μF (100 nF), 10-25V capacitor is recommended. The capacitor should be a low-ESR device, with a resonance frequency in the range of 200 MHz and higher. Ceramic capacitors are recommended.
- **Placement on the printed circuit board:** The decoupling capacitors should be placed as close to the pins as possible. It is recommended to place the capacitors on the same side of the board as the device. If space is constricted, the capacitor can be placed on another layer on the PCB using a via; however, ensure that the trace length from the pin to the capacitor is no greater than 0.25 inch (6 mm).
- **Handling high-frequency noise:** If the board is experiencing high-frequency noise (upward of tens of MHz), add a second ceramic type capacitor in parallel to the above described decoupling capacitor. The value of the second capacitor can be in the range of 0.01 μF to 0.001 μF. Place this second capacitor next to each primary decoupling capacitor. In high-speed circuit designs, consider implementing a decade pair of capacitances as close to the power and ground pins as possible (e.g., 0.1 μF in parallel with 0.001 μF).
- **Maximizing performance:** On the board layout from the power supply circuit, run the power and return traces to the decoupling capacitors first, and then to the device pins. This ensures that the decoupling capacitors are first in the power chain. Equally important is to keep the trace length between the capacitor and the power pins to a minimum, thereby reducing PCB trace inductance.

### 2.2.2 TANK CAPACITORS

On boards with power traces running longer than six inches in length, it is suggested to use a tank capacitor for integrated circuits, including microcontrollers, to supply a local power source. The value of the tank capacitor should be determined based on the trace resistance that connects the power supply source to the device, and the maximum current drawn by the device in the application. In other words, select the tank capacitor so that it meets the acceptable voltage sag at the device. Typical values range from 4.7 μF to 47 μF.

### 4.1.1 READING PROGRAM MEMORY AS DATA

There are three methods of accessing constants in program memory. The first method is to use tables of RETLW instructions. The second method is to set an FSR to point to the program memory. The third method is to use the NVMREG interface to access the program memory. For an example of NVMREG interface use, reference Section 13.3, NVMREG Access.

#### 4.1.1.1 RETLW Instruction

The RETLW instruction can be used to provide access to tables of constants. The recommended way to create such a table is shown in Example 4-1.

**EXAMPLE 4-1: RETLW INSTRUCTION**

```
constants
    BRW                 ;Add Index in W to
                        ;program counter to
                        ;select data
    RETLW DATA0         ;Index0 data
    RETLW DATA1         ;Index1 data
    RETLW DATA2
    RETLW DATA3


my_function
    ;… LOTS OF CODE…
    MOVLW       DATA_INDEX
    call constants
    ;… THE CONSTANT IS IN W
```

The BRW instruction makes this type of table very simple to implement.

#### 4.1.1.2 Indirect Read with FSR

The program memory can be accessed as data by setting bit 7 of an FSRxH register and reading the matching INDFx register. The MOVIW instruction will place the lower eight bits of the addressed word in the W register. Writes to the program memory cannot be performed via the INDF registers. Instructions that read the program memory via the FSR require one extra instruction cycle to complete. Example 4-2 demonstrates reading the program memory via an FSR.

**TABLE 4-11: SPECIAL FUNCTION REGISTER SUMMARY BANKS 0-63 (CONTINUED)**

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on: POR, BOR | Value on: MCLR |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Bank 20** | | | | | | | | | | | |
| CPU CORE REGISTERS; see Table 4-3 for specifics | | | | | | | | | | | |
| A0Ch — A18h | — | Unimplemented | | | | | | | | — | — |
| A19h | RC2REG | RC2REG<7:0> | | | | | | | | 0000 0000 | 0000 0000 |
| A1Ah | TX2REG | TX2REG<7:0> | | | | | | | | 0000 0000 | 0000 0000 |
| A1Bh | SP2BRGL | SP2BRGL<7:0> | | | | | | | | 0000 0000 | 0000 0000 |
| A1Ch | SP2BRGH | SP2BRGH<7:0> | | | | | | | | 0000 0000 | 0000 0000 |
| A1Dh | RC2STA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 0000 0000 | 0000 0000 |
| A1Eh | TX2STA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 0000 0010 | 0000 0010 |
| A1Fh | BAUD2CON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 01-0 0-00 | 01-0 0-00 |

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

**TABLE 5-1: BOOT BLOCK SIZE BITS**

| $\overline{\text{BBEN}}$ | BBSIZE<2:0> | Actual Boot Block Size<br>User Program Memory Size (words) | | Last Boot Block Memory Access |
|---|---|---|---|---|
| | | PIC16(L)F15375/85 | PIC16(L)F15356/76/86 | |
| 1 | xxx | 0 | 0 | — |
| 0 | 111 | 512 | 512 | 01FFh |
| 0 | 110 | 1024 | 1024 | 03FFh |
| 0 | 101 | 2048 | 2048 | 07FFh |
| 0 | 100 | 4096 | 4096 | 0FFFh |
| 0 | 011 | — | 8192 | 1FFFh |

**Note 1:** The maximum boot block size is half the user program memory size. All selections higher than the maximum are set to half size. For example, all BBSIZE = 000 - 100 produce a boot block size of 4kW on a 8kW device.

**REGISTER 5-5: CONFIGURATION WORD 5: CODE PROTECTION**

| U-1 | U-1 | U-1 | U-1 | U-1 | U-1 |
|---|---|---|---|---|---|
| — | — | — | — | — | — |
| bit 13 | | | | | bit 8 |

| U-1 | U-1 | U-1 | U-1 | U-1 | U-1 | U-1 | R/P-1 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | — | $\overline{\text{CP}}$ |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | P = Programmable bit | x = Bit is unknown | U = Unimplemented bit, read as ‘1’ |
| ‘0’ = Bit is cleared | ‘1’ = Bit is set | W = Writable bit | n = Value when blank or after Bulk Erase |

bit 13-1 **Unimplemented:** Read as ‘1’

bit 0 **$\overline{\text{CP}}$:** Program Flash Memory Code Protection bit
1 = Program Flash Memory code protection disabled
0 = Program Flash Memory code protection enabled

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

The Low-Power Brown-out Reset (LPBOR) is an important part of the Reset subsystem. Refer to Figure 8-1 to see how the BOR and LPBOR interact with other modules.

The LPBOR is used to monitor the external VDD pin. When too low of a voltage is detected, the device is held in Reset.

### 8.4.1 ENABLING LPBOR

The LPBOR is controlled by the $\overline{\text{LPBOR}}$ bit of the Configuration Word (Register 5-1). When the device is erased, the LPBOR module defaults to disabled.

### 8.4.2 LPBOR MODULE OUTPUT

The output of the LPBOR module is a signal indicating whether or not a Reset is to be asserted. When this occurs, a register bit ($\overline{\text{BOR}}$) is changed to indicate that a BOR Reset has occurred. The same bit is set for either the BOR or the LPBOR (refer to Register 8-3). This signal is OR'd with the output of the BOR module to provide the generic $\overline{\text{BOR}}$ signal, which goes to the PCON register and to the power control block. Refer to Figure 8-1 for the OR gate connections of the BOR and LPBOR Reset signals, which eventually generates one common BOR Reset.

## 8.5 $\overline{\text{MCLR}}$

The $\overline{\text{MCLR}}$ is an optional external input that can reset the device. The $\overline{\text{MCLR}}$ function is controlled by the MCLRE bit of Configuration Words and the LVP bit of Configuration Words (Table 8-2).

**TABLE 8-2: $\overline{\text{MCLR}}$ CONFIGURATION**

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

### 8.5.1 $\overline{\text{MCLR}}$ 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. Refer to **Section 2.3 "Master Clear (MCLR) Pin"** for recommended $\overline{\text{MCLR}}$ connections.

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.

### 8.5.2 $\overline{\text{MCLR}}$ 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 14.1 "I/O Priorities"** for more information.

## 8.6 Windowed Watchdog Timer (WWDT) Reset

The Watchdog Timer generates a Reset if the firmware does not issue a CLRWDT instruction within the time-out period and the window is open. The $\overline{\text{TO}}$ and $\overline{\text{PD}}$ bits in the STATUS register and the $\overline{\text{WDT}}$ bit in PCON are changed to indicate a WDT Reset caused by the timer overflowing, and WDTWV bit in the PCON register is changed to indicate a WDT Reset caused by a window violation. See **Section 12.0 "Windowed Watchdog Timer (WWDT)"** for more information.

## 8.7 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 8-4 for default conditions after a RESET instruction has occurred.

## 8.8 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 Configuration Words. See **Section 4.5.2 "Overflow/Underflow Reset"** for more information.

## 8.9 Programming Mode Exit

Upon exit of In-Circuit Serial Programming™ (ICSP™) mode, the device will behave as if a POR had just occurred (the device does not reset upon run time self-programming/erase operations).

## 8.10 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.

The Power-up Timer provides a nominal 64 ms time out on POR or Brown-out Reset. The device is held in Reset as long as PWRT is active. The PWRT delay allows additional time for the VDD to rise to an acceptable level. The Power-up Timer is enabled by clearing the $\overline{\text{PWRTE}}$ bit in the Configuration Words. The Power-up Timer starts after the release of the POR and BOR. For additional information, refer to Application Note AN607, "*Power-up Trouble Shooting*" (DS00607).

**TABLE 15-6: PPS OUTPUT SIGNAL ROUTING OPTIONS (PIC16(L)F15375/76)**

| Output Signal Name | RxyPPS Register Value | Remappable to Pins of PORTx | | | | |
|---|---|---|---|---|---|---|
| | | PIC16(L)F15375/76 | | | | |
| | | PORTA | PORTB | PORTC | PORTD | PORTE |
| CLKR | 0x1B | | • | • | | |
| NCO1OUT | 0x1A | • | | | • | |
| TMR0 | 0x19 | | • | • | | |
| SDO2/SDA2 | 0x18 | | • | | • | |
| SCK2/SCL2 | 0x17 | | • | | • | |
| SDO1/SDA1 | 0x16 | | • | • | | |
| SCK1/SCL1 | 0x15 | | • | • | | |
| C2OUT | 0x14 | • | | | | • |
| C1OUT | 0x13 | • | | | • | |
| DT2 | 0x12 | | • | | • | |
| TX2/CK2 | 0x11 | | • | | • | |
| DT1 | 0x10 | | • | • | | |
| TX1/CK1 | 0x0F | | • | • | | |
| PWM6OUT | 0x0E | • | | | • | |
| PWM5OUT | 0x0D | • | | • | | |
| PWM4OUT | 0x0C | | • | | • | |
| PWM3OUT | 0x0B | | • | | • | |
| CCP2 | 0x0A | | • | • | | |
| CCP1 | 0x09 | | • | • | | |
| CWG1D | 0x08 | | • | | • | |
| CWG1C | 0x07 | | • | | • | |
| CWG1B | 0x06 | | • | | • | |
| CWG1A | 0x05 | | • | • | | |
| CLC4OUT | 0x04 | | • | | • | |
| CLC3OUT | 0x03 | | • | | • | |
| CLC2OUT | 0x02 | • | | • | | |
| CLC1OUT | 0x01 | • | | • | | |

**TABLE 21-1: SUMMARY OF REGISTERS ASSOCIATED WITH THE DAC1 MODULE**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|---|---|---|---|---|---|---|---|---|---|
| DAC1CON0 | DAC1EN | — | DAC1OE1 | DAC1OE2 | DAC1PSS<1:0> | | — | DAC1NSS | 287 |
| DAC1CON1 | — | — | — | DAC1R<4:0> | | | | | 287 |
| CM1PSEL | — | — | — | — | — | PCH<2:0> | | | 307 |
| CM2PSEL | — | — | — | — | — | PCH<2:0> | | | 307 |

**Legend:** — = Unimplemented location, read as '0'. Shaded cells are not used with the DAC module.

## 23.2 Comparator Control

Each comparator has two control registers: CMxCON0 and CMxCON1.

The CMxCON0 register (see Register 23-1) contains Control and Status bits for the following:

- Enable
- Output
- Output polarity
- Hysteresis enable
- Timer1 output synchronization

The CMxCON1 register (see Register 23-2) contains Control bits for the following:

- Interrupt on positive/negative edge enables
- The CMxNSEL and CMxPSEL (Register 23-3 and Register 23-4) contain control bits for the following:
  - Positive input channel selection
  - Negative input channel selection

### 23.2.1 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.

### 23.2.2 COMPARATOR OUTPUT

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.

The comparator output can also be routed to an external pin through the RxyPPS register (Register 15-2). The corresponding TRIS bit must be clear to enable the pin as an output.

| | |
|---|---|
| **Note 1:** | The internal output of the comparator is latched with each instruction cycle. Unless otherwise specified, external outputs are not latched. |

### 23.2.3 COMPARATOR OUTPUT POLARITY

Inverting the output of the comparator is functionally equivalent to swapping the comparator inputs. The polarity of the comparator output can be inverted by setting the CxPOL bit of the CMxCON0 register. Clearing the CxPOL bit results in a non-inverted output.

Table 23-2 shows the output state versus input conditions, including polarity control.

**TABLE 23-2: COMPARATOR OUTPUT STATE VS. INPUT CONDITIONS**

| Input Condition | CxPOL | CxOUT |
|---|---|---|
| CxVN > CxVP | 0 | 0 |
| CxVN < CxVP | 0 | 1 |
| CxVN > CxVP | 1 | 1 |
| CxVN < CxVP | 1 | 0 |

**REGISTER 23-3: CMxNSEL: COMPARATOR Cx NEGATIVE INPUT SELECT REGISTER**

| U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | NCH<2:0> | | |
| 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 7-3 **Unimplemented:** Read as '0'

bit 2-0 **NCH<2:0>:** Comparator Negative Input Channel Select bits

111 = CxVN connects to AVSS
110 = CxVN connects to FVR Buffer 2
101 = CxVN unconnected
100 = CxVN unconnected
011 = CxVN connects to CxIN3- pin
010 = CxVN connects to CxIN2- pin
001 = CxVN connects to CxIN1- pin
000 = CxVN connects to CxIN0- pin

**REGISTER 23-4: CMxPSEL: COMPARATOR Cx POSITIVE INPUT SELECT REGISTER**

| U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | PCH<2:0> | | |
| 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 7-3 **Unimplemented:** Read as '0'

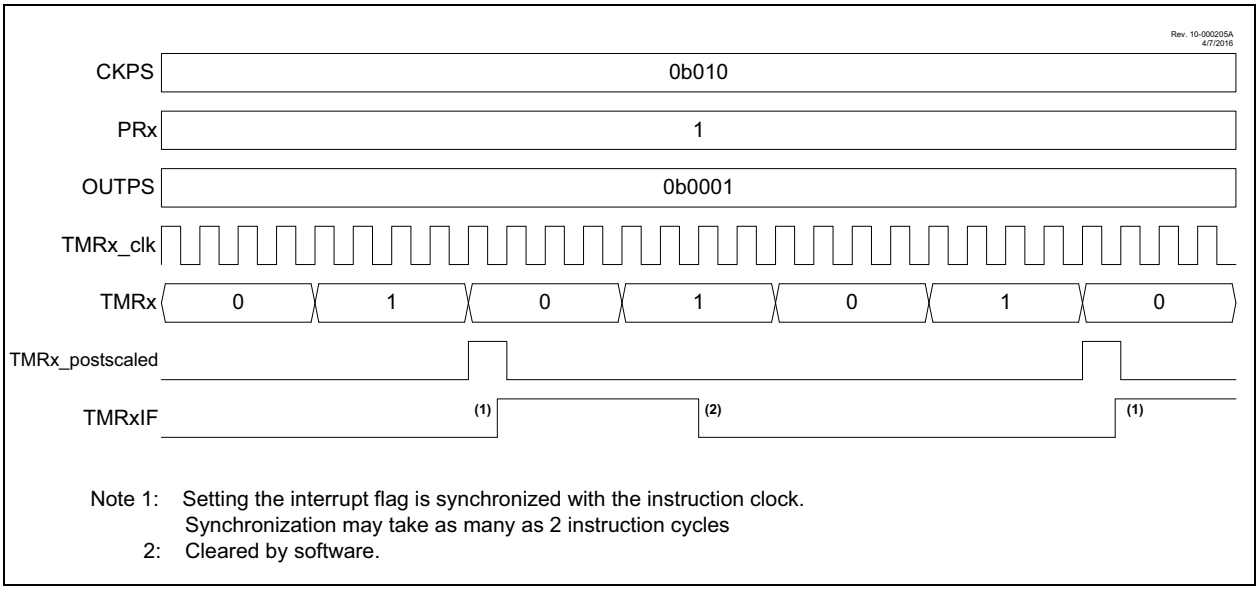
bit 2-0 **PCH<2:0>:** Comparator Positive Input Channel Select bits

111 = CxVP connects to AVSS
110 = CxVP connects to FVR Buffer 2
101 = CxVP connects to DAC output
100 = CxVP unconnected
011 = CxVP unconnected
010 = CxVP unconnected
001 = CxVP connects to CxIN1+ pin
000 = CxVP connects to CxIN0+ pin

## 27.4 Timer2 Interrupt

Timer2 can also generate a device interrupt. The interrupt is generated when the postscaler counter matches one of 16 postscale options (from 1:1 through 1:16), which are selected with the postscaler control bits, OUTPS<3:0> of the T2CON register. The interrupt is enabled by setting the TMR2IE interrupt enable bit of the PIE4 register. Interrupt timing is illustrated in Figure 27-3.

**FIGURE 27-3: TIMER2 PRESCALER, POSTSCALER, AND INTERRUPT TIMING DIAGRAM**




Note 1: Setting the interrupt flag is synchronized with the instruction clock. Synchronization may take as many as 2 instruction cycles
2: Cleared by software.

## 31.0 CONFIGURABLE LOGIC CELL (CLC)

The Configurable Logic Cell (CLCx) module provides programmable logic that operates outside the speed limitations of software execution. The logic cell selects from 40 input signals and, through the use of configurable gates, reduces the inputs to four logic lines that drive one of eight selectable single-output logic functions.

Input sources are a combination of the following:

- I/O pins
- Internal clocks
- Peripherals
- Register bits

The output can be directed internally to peripherals and to an output pin.

The CLC modules available are shown in Table 31-1.

**TABLE 31-1: AVAILABLE CLC MODULES**

| Device | CLC1 | CLC2 | CLC3 | CLC4 |
|---|---|---|---|---|
| PIC16(L)F15356/75/76/85/86 | • | • | • | • |

**Note:** The CLC1, CLC2, CLC3 and CLC4 are four separate module instances of the same CLC module design. Throughout this section, the lower case 'x' in register and bit names is a generic reference to the CLC number (which should be substituted with 1, 2, 3, or 4 during code development). For example, the control register is generically described in this chapter as CLCxCON, but the actual device registers are CLC1CON, CLC2CON, CLC3CON and CLC4CON. Similarly, the LCxEN bit represents the LC1EN, LC2EN, LC3EN and LC4EN bits.

Refer to Figure 31-1 for a simplified diagram showing signal flow through the CLCx.

Possible configurations include:

- Combinatorial Logic
  - AND
  - NAND
  - AND-OR
  - AND-OR-INVERT
  - OR-XOR
  - OR-XNOR
- Latches
  - S-R
  - Clocked D with Set and Reset
  - Transparent D with Set and Reset
  - Clocked J-K with Reset

The $I^2C$ interface supports the following modes and features:

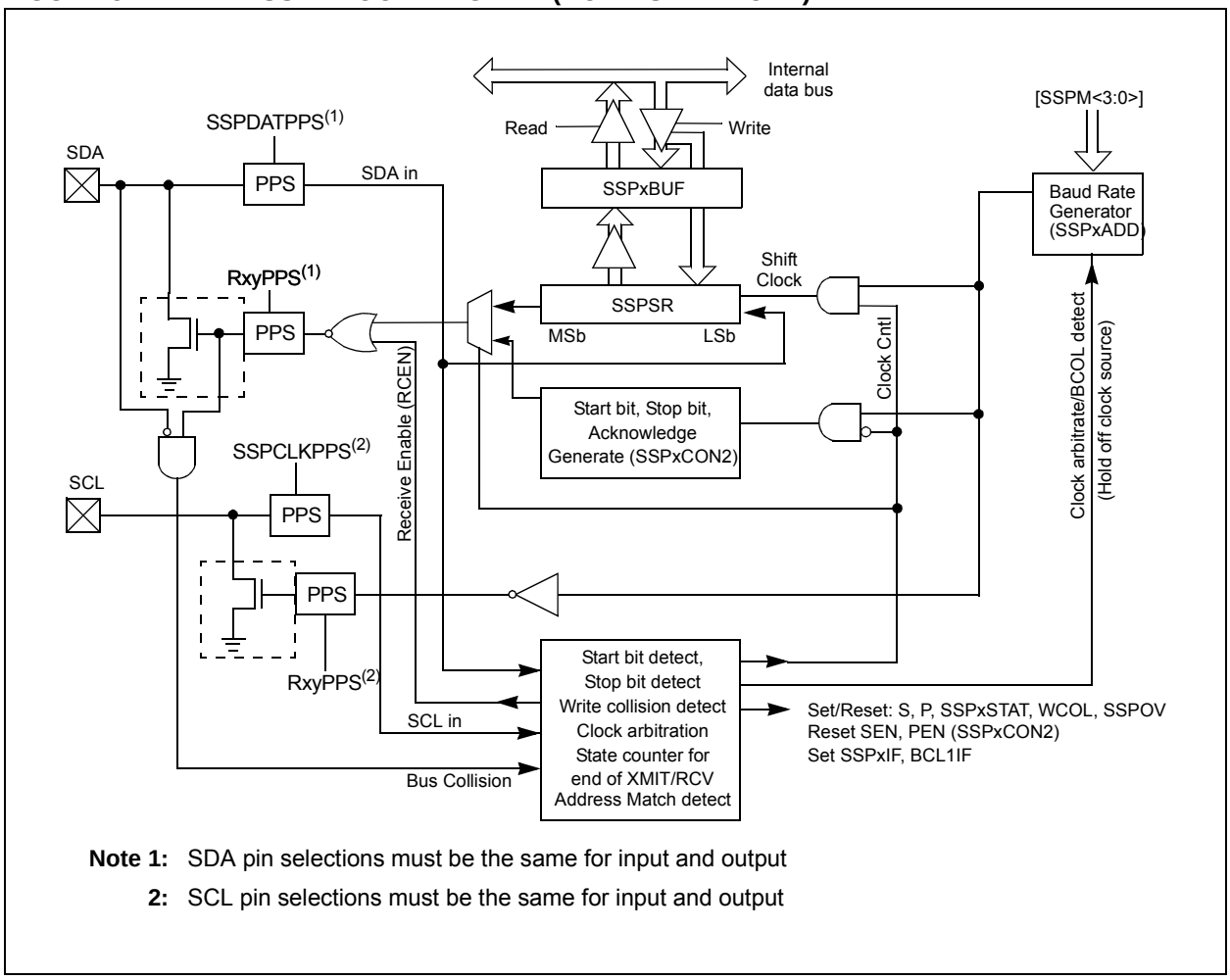
- Master mode
- Slave mode
- Byte NACKing (Slave mode)
- Limited multi-master support
- 7-bit and 10-bit addressing
- Start and Stop interrupts
- Interrupt masking
- Clock stretching
- Bus collision detection
- General call address matching
- Address masking
- Selectable SDA hold times

Figure 32-2 is a block diagram of the $I^2C$ interface module in Master mode. Figure 32-3 is a diagram of the $I^2C$ interface module in Slave mode.

**Note 1:** In devices with more than one MSSP module, it is very important to pay close attention to SSPxCONx register names. SSPxCON1 and SSPxCON2 registers control different operational aspects of the same module, while SSP1CON1 and SSP2CON1 control the same features for two different modules.

**2:** Throughout this section, generic references to an MSSPx module in any of its operating modes may be interpreted as being equally applicable to MSSP1 or MSSP2. Register names, module I/O signals, and bit names may use the generic designator 'x' to indicate the use of a numeral to distinguish a particular module when required.

**FIGURE 32-2: MSSP BLOCK DIAGRAM ($I^2C$ MASTER MODE)**




**Note 1:** SDA pin selections must be the same for input and output

**2:** SCL pin selections must be the same for input and output

### 32.5.4 SLAVE MODE 10-BIT ADDRESS RECEPTION

This section describes a standard sequence of events for the MSSP module configured as an $I^2C$ slave in 10-bit Addressing mode.

Figure 32-20 is used as a visual reference for this description.

This is a step by step process of what must be done by slave software to accomplish $I^2C$ communication.

1. Bus starts Idle.
2. Master sends Start condition; S bit of SSPxSTAT is set; SSPxIF is set if interrupt on Start detect is enabled.
3. Master sends matching high address with R/$\overline{W}$ bit clear; UA bit of the SSPxSTAT register is set.
4. Slave sends $\overline{ACK}$ and SSPxIF is set.
5. Software clears the SSPxIF bit.
6. Software reads received address from SSPxBUF clearing the BF flag.
7. Slave loads low address into SSPxADD, releasing SCL.
8. Master sends matching low address byte to the slave; UA bit is set.

**Note:** Updates to the SSPxADD register are not allowed until after the $\overline{ACK}$ sequence.

9. Slave sends $\overline{ACK}$ and SSPxIF is set.

**Note:** If the low address does not match, SSPxIF and UA are still set so that the slave software can set SSPxADD back to the high address. BF is not set because there is no match. CKP is unaffected.

10. Slave clears SSPxIF.
11. Slave reads the received matching address from SSPxBUF clearing BF.
12. Slave loads high address into SSPxADD.
13. Master clocks a data byte to the slave and clocks out the slaves $\overline{ACK}$ on the ninth SCL pulse; SSPxIF is set.
14. If SEN bit of SSPxCON2 is set, CKP is cleared by hardware and the clock is stretched.
15. Slave clears SSPxIF.
16. Slave reads the received byte from SSPxBUF clearing BF.
17. If SEN is set the slave sets CKP to release the SCL.
18. Steps 13-17 repeat for each received byte.
19. Master sends Stop to end the transmission.

### 32.5.5 10-BIT ADDRESSING WITH ADDRESS OR DATA HOLD

Reception using 10-bit addressing with AHEN or DHEN set is the same as with 7-bit modes. The only difference is the need to update the SSPxADD register using the UA bit. All functionality, specifically when the CKP bit is cleared and SCL line is held low are the same. Figure 32-21 can be used as a reference of a slave in 10-bit addressing with AHEN set.

Figure 32-22 shows a standard waveform for a slave transmitter in 10-bit Addressing mode.

## REGISTER 32-2: SSPxCON1: SSPx CONTROL REGISTER 1

| R/C/HS-0/0 | R/C/HS-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 |
|---|---|---|---|---|---|---|---|
| WCOL | SSPOV(1) | SSPEN | CKP | SSPM<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 | HS = Bit is set by hardware | C = User cleared |

bit 7 **WCOL:** Write Collision Detect bit (Transmit mode only)
1 = The SSPxBUF register is written while it is still transmitting the previous word (must be cleared in software)
0 = No collision

bit 6 **SSPOV:** Receive Overflow Indicator bit(1)
In SPI mode:
1 = A new byte is received while the SSPxBUF register is still holding the previous data. In case of overflow, the data in SSPxSR is lost. Overflow can only occur in Slave mode. In Slave mode, the user must read the SSPxBUF, even if only transmitting data, to avoid setting overflow. In Master mode, the overflow bit is not set since each new reception (and transmission) is initiated by writing to the SSPxBUF register (must be cleared in software).
0 = No overflow
In $I^2C$ mode:
1 = A byte is received while the SSPxBUF register is still holding the previous byte. SSPOV is a "don't care" in Transmit mode (must be cleared in software).
0 = No overflow

bit 5 **SSPEN:** Synchronous Serial Port Enable bit
In both modes, when enabled, the following pins must be properly configured as input or output
In SPI mode:
1 = Enables serial port and configures SCK, SDO, SDI and $\overline{SS}$ as the source of the serial port pins(2)
0 = Disables serial port and configures these pins as I/O port pins
In $I^2C$ mode:
1 = Enables the serial port and configures the SDA and SCL pins as the source of the serial port pins(3)
0 = Disables serial port and configures these pins as I/O port pins

bit 4 **CKP:** Clock Polarity Select bit
In SPI mode:
1 = Idle state for clock is a high level
0 = Idle state for clock is a low level
In $I^2C$ Slave mode:
SCL release control
1 = Enable clock
0 = Holds clock low (clock stretch). (Used to ensure data setup time.)
In $I^2C$ Master mode:
Unused in this mode

bit 3-0 **SSPM<3:0>:** Synchronous Serial Port Mode Select bits
1111 = $I^2C$ Slave mode, 10-bit address with Start and Stop bit interrupts enabled
1110 = $I^2C$ Slave mode, 7-bit address with Start and Stop bit interrupts enabled
1101 = Reserved
1100 = Reserved
1011 = $I^2C$ firmware controlled Master mode (slave idle)
1010 = SPI Master mode, clock = FOSC/(4 * (SSPxADD+1))(5)
1001 = Reserved
1000 = $I^2C$ Master mode, clock = FOSC / (4 * (SSPxADD+1))(4)
0111 = $I^2C$ Slave mode, 10-bit address
0110 = $I^2C$ Slave mode, 7-bit address
0101 = SPI Slave mode, clock = SCK pin, $\overline{SS}$ pin control disabled, $\overline{SS}$ can be used as I/O pin
0100 = SPI Slave mode, clock = SCK pin, $\overline{SS}$ pin control enabled
0011 = SPI Master mode, clock = T2_match/2
0010 = SPI Master mode, clock = FOSC/64
0001 = SPI Master mode, clock = FOSC/16
0000 = SPI Master mode, clock = FOSC/4

**Note 1:** In Master mode, the overflow bit is not set since each new reception (and transmission) is initiated by writing to the SSPxBUF register.
**2:** When enabled, these pins must be properly configured as input or output. Use SSPxSSPPS, SSPxCLKPPS, SSPxDATPPS, and RxyPPS to select the pins.
**3:** When enabled, the SDA and SCL pins must be configured as inputs. Use SSPxCLKPPS, SSPxDATPPS, and RxyPPS to select the pins.
**4:** SSPxADD values of 0, 1 or 2 are not supported for $I^2C$ mode.
**5:** SSPxADD value of '0' is not supported. Use SSPM = 0000 instead.

### 33.3.5 RECEIVING A BREAK CHARACTER

The Enhanced EUSART module can receive a Break character in two ways.

The first method to detect a Break character uses the FERR bit of the RCxSTA register and the received data as indicated by RCxREG. The Baud Rate Generator is assumed to have been initialized to the expected baud rate.

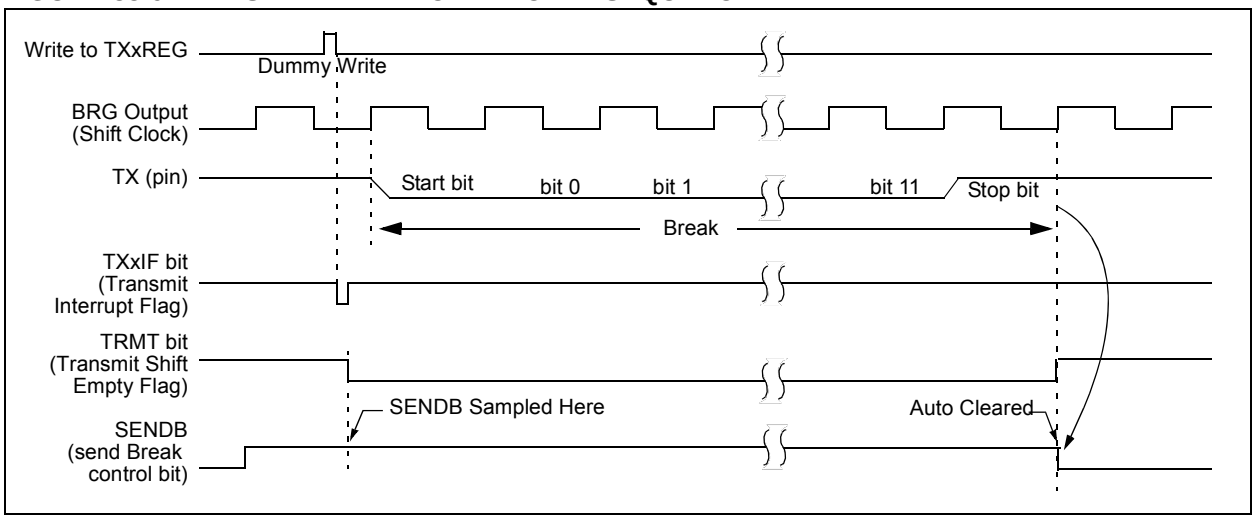
A Break character has been received when:

- RXxIF bit is set
- FERR bit is set
- RCxREG = 00h

The second method uses the Auto-Wake-up feature described in **Section 33.3.3 "Auto-Wake-up on Break"**. By enabling this feature, the EUSART will sample the next two transitions on RX/DT, cause an RXxIF interrupt, and receive the next data byte followed by another interrupt.

Note that following a Break character, the user will typically want to enable the Auto-Baud Detect feature. For both methods, the user can set the ABDEN bit of the BAUDxCON register before placing the EUSART in Sleep mode.

**FIGURE 33-9: SEND BREAK CHARACTER SEQUENCE**



#### 33.4.2.3 EUSART Synchronous Slave Reception

The operation of the Synchronous Master and Slave modes is identical (**Section 33.4.1.5 "Synchronous Master Reception"**), with the following exceptions:

- Sleep
- CREN bit is always set, therefore the receiver is never idle
- SREN bit, which is a "don't care" in Slave mode

A character may be received while in Sleep mode by setting the CREN bit prior to entering Sleep. Once the word is received, the RSR register will transfer the data to the RCxREG register. If the RXxIE enable bit is set, the interrupt generated will wake the device from Sleep and execute the next instruction. If the GIE bit is also set, the program will branch to the interrupt vector.

#### 33.4.2.4 Synchronous Slave Reception Set-up:

1. Set the SYNC and SPEN bits and clear the CSRC bit.
2. Clear the ANSEL bit for both the CK and DT pins (if applicable).
3. If interrupts are desired, set the RXxIE bit of the PIE3 register and the GIE and PEIE bits of the INTCON register.
4. If 9-bit reception is desired, set the RX9 bit.
5. Set the CREN bit to enable reception.
6. The RXxIF bit will be set when reception is complete. An interrupt will be generated if the RXxIE bit was set.
7. If 9-bit mode is enabled, retrieve the Most Significant bit from the RX9D bit of the RCxSTA register.
8. Retrieve the eight Least Significant bits from the receive FIFO by reading the RCxREG register.
9. If an overrun error occurs, clear the error by either clearing the CREN bit of the RCxSTA register or by clearing the SPEN bit which resets the EUSART.

**REGISTER 33-2: RCxSTA: RECEIVE STATUS AND CONTROL REGISTER**

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R-0/0 | R-0/0 | R-0/0 |
|---|---|---|---|---|---|---|---|
| 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 RCxREG 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 EUSART 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.

**TABLE 33-3: BAUD RATE FORMULAS**

| Configuration Bits | | | BRG/EUSART Mode | Baud Rate Formula |
|---|---|---|---|---|
| SYNC | BRG16 | BRGH | | |
| 0 | 0 | 0 | 8-bit/Asynchronous | FOSC/[64 (n+1)] |
| 0 | 0 | 1 | 8-bit/Asynchronous | FOSC/[16 (n+1)] |
| 0 | 1 | 0 | 16-bit/Asynchronous | |
| 0 | 1 | 1 | 16-bit/Asynchronous | FOSC/[4 (n+1)] |
| 1 | 0 | x | 8-bit/Synchronous | |
| 1 | 1 | x | 16-bit/Synchronous | |

**Legend:** x = Don't care, n = value of SPxBRGH, SPxBRGL register pair.

**TABLE 33-4: BAUD RATE FOR ASYNCHRONOUS MODES**

| BAUD RATE | SYNC = 0, BRGH = 0, BRG16 = 0 | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FOSC = 32.000 MHz | | | FOSC = 20.000 MHz | | | FOSC = 18.432 MHz | | | FOSC = 11.0592 MHz | | |
| | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) |
| 300 | — | — | — | — | — | — | — | — | — | — | — | — |
| 1200 | — | — | — | 1221 | 1.73 | 255 | 1200 | 0.00 | 239 | 1200 | 0.00 | 143 |
| 2400 | 2404 | 0.16 | 207 | 2404 | 0.16 | 129 | 2400 | 0.00 | 119 | 2400 | 0.00 | 71 |
| 9600 | 9615 | 0.16 | 51 | 9470 | -1.36 | 32 | 9600 | 0.00 | 29 | 9600 | 0.00 | 17 |
| 10417 | 10417 | 0.00 | 47 | 10417 | 0.00 | 29 | 10286 | -1.26 | 27 | 10165 | -2.42 | 16 |
| 19.2k | 19.23k | 0.16 | 25 | 19.53k | 1.73 | 15 | 19.20k | 0.00 | 14 | 19.20k | 0.00 | 8 |
| 57.6k | 55.55k | -3.55 | 3 | — | — | — | 57.60k | 0.00 | 7 | 57.60k | 0.00 | 2 |
| 115.2k | — | — | — | — | — | — | — | — | — | — | — | — |

| BAUD RATE | SYNC = 0, BRGH = 0, BRG16 = 0 | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FOSC = 8.000 MHz | | | FOSC = 4.000 MHz | | | FOSC = 3.6864 MHz | | | FOSC = 1.000 MHz | | |
| | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) |
| 300 | — | — | — | 300 | 0.16 | 207 | 300 | 0.00 | 191 | 300 | 0.16 | 51 |
| 1200 | 1202 | 0.16 | 103 | 1202 | 0.16 | 51 | 1200 | 0.00 | 47 | 1202 | 0.16 | 12 |
| 2400 | 2404 | 0.16 | 51 | 2404 | 0.16 | 25 | 2400 | 0.00 | 23 | — | — | — |
| 9600 | 9615 | 0.16 | 12 | — | — | — | 9600 | 0.00 | 5 | — | — | — |
| 10417 | 10417 | 0.00 | 11 | 10417 | 0.00 | 5 | — | — | — | — | — | — |
| 19.2k | — | — | — | — | — | — | 19.20k | 0.00 | 2 | — | — | — |
| 57.6k | — | — | — | — | — | — | 57.60k | 0.00 | 0 | — | — | — |
| 115.2k | — | — | — | — | — | — | — | — | — | — | — | — |

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

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

Operands: n ∈ [0,1]
mm ∈ [00,01, 10, 11]
-32 ≤ k ≤ 31

Operation: W → 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 wrap-around.

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
```

**RESET** **Software Reset**

Syntax: [ *label* ] RESET

Operands: None

Operation: Execute a device Reset. Resets the $\overline{RI}$ flag of the PCON register.

Status Affected: None

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

**RETFIE** **Return from Interrupt**

Syntax: [ *label* ] RETFIE k

Operands: None

Operation: TOS → PC,
1 → GIE

Status Affected: None

Description: Return from Interrupt. Stack is POPed and Top-of-Stack (TOS) is loaded in the PC. Interrupts are enabled by setting Global Interrupt Enable bit, GIE (INTCON<7>). This is a 2-cycle instruction.

Words: 1

Cycles: 2

Example:

```
RETFIE
```

After Interrupt
PC = TOS
GIE = 1