**Welcome to E-XFL.COM**

## What is "Embedded - Microcontrollers"?

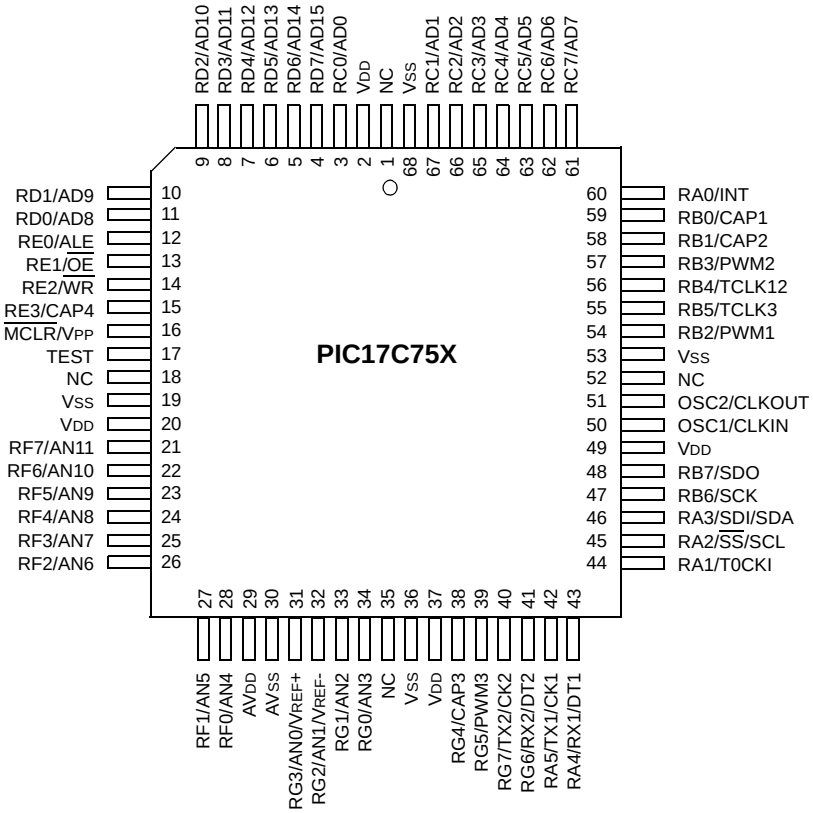
"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" 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 | 16MHz |
| Connectivity | I²C, SPI, UART/USART |
| Peripherals | Brown-out Detect/Reset, POR, PWM, WDT |
| Number of I/O | 66 |
| Program Memory Size | 32KB (16K x 16) |
| Program Memory Type | OTP |
| EEPROM Size | - |
| RAM Size | 902 x 8 |
| Voltage - Supply (Vcc/Vdd) | 4.5V ~ 5.5V |
| Data Converters | A/D 16x10b |
| Oscillator Type | External |
| Operating Temperature | 0°C ~ 70°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 84-LCC (J-Lead) |
| Supplier Device Package | 84-PLCC (29.31x29.31) |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/pic17c766-16-l |

## Pin Diagrams cont.'d

### 68-Pin PLCC




### 64-Pin TQFP

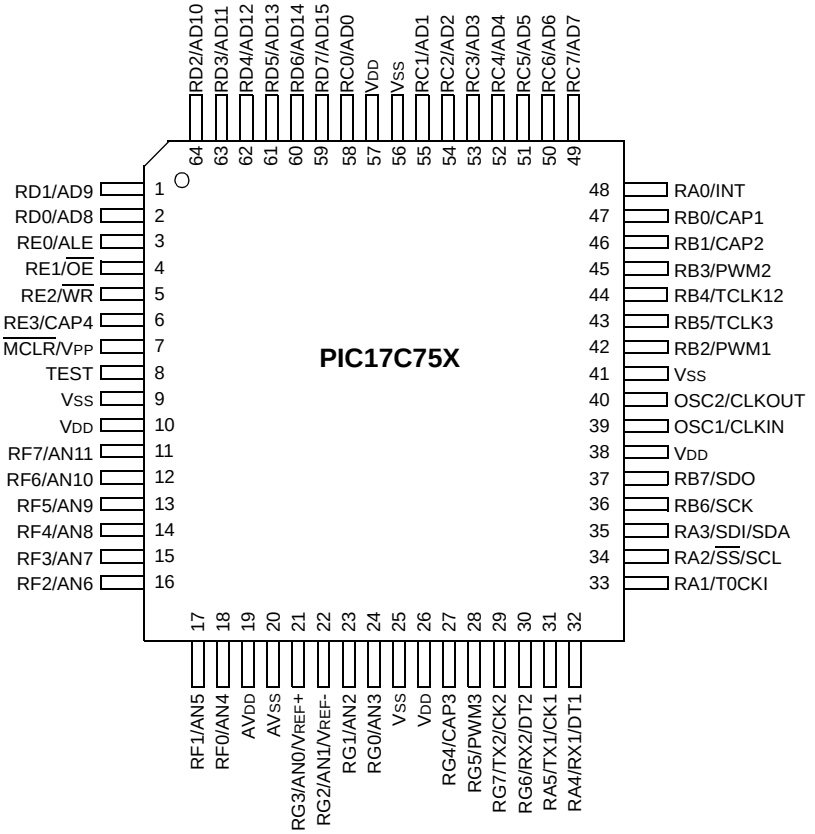


**TABLE 3-1: PINOUT DESCRIPTIONS**

| Name | PIC17C75X | | | PIC17C76X | | I/O/P Type | Buffer Type | Description |
|---|---|---|---|---|---|---|---|---|
| | DIP No. | PLCC No. | TQFP No. | PLCC No. | QFP No. | | | |
| OSC1/CLKIN | 47 | 50 | 39 | 62 | 49 | I | ST | Oscillator input in Crystal/Resonator or RC Oscillator mode. External clock input in External Clock mode. |
| OSC2/CLKOUT | 48 | 51 | 40 | 63 | 50 | O | — | Oscillator output. Connects to crystal or resonator in Crystal Oscillator mode. In RC Oscillator or External Clock modes, OSC2 pin outputs CLKOUT which has one fourth the frequency (FOSC/4) of OSC1 and denotes the instruction cycle rate. |
| $\overline{MCLR}$/VPP | 15 | 16 | 7 | 20 | 9 | I/P | ST | Master clear (RESET) input or Programming Voltage (VPP) input. This is the active low RESET input to the device. |
| | | | | | | | | PORTA pins have individual differentiations that are listed in the following descriptions: |
| RA0/INT | 56 | 60 | 48 | 72 | 58 | I | ST | RA0 can also be selected as an external interrupt input. Interrupt can be configured to be on positive or negative edge. Input only pin. |
| RA1/T0CKI | 41 | 44 | 33 | 56 | 43 | I | ST | RA1 can also be selected as an external interrupt input and the interrupt can be configured to be on positive or negative edge. RA1 can also be selected to be the clock input to the Timer0 timer/counter. Input only pin. |
| RA2/$\overline{SS}$/SCL | 42 | 45 | 34 | 57 | 44 | I/O(2) | ST | RA2 can also be used as the slave select input for the SPI or the clock input for the $I^2C$ bus. High voltage, high current, open drain port pin. |
| RA3/SDI/SDA | 43 | 46 | 35 | 58 | 45 | I/O(2) | ST | RA3 can also be used as the data input for the SPI or the data for the $I^2C$ bus. High voltage, high current, open drain port pin. |
| RA4/RX1/DT1 | 40 | 43 | 32 | 51 | 38 | I/O(1) | ST | RA4 can also be selected as the USART1 (SCI) Asynchronous Receive or USART1 (SCI) Synchronous Data. Output available from USART only. |
| RA5/TX1/CK1 | 39 | 42 | 31 | 50 | 37 | I/O(1) | ST | RA5 can also be selected as the USART1 (SCI) Asynchronous Transmit or USART1 (SCI) Synchronous Clock. Output available from USART only. |
| | | | | | | | | PORTB is a bi-directional I/O Port with software configurable weak pull-ups. |
| RB0/CAP1 | 55 | 59 | 47 | 71 | 57 | I/O | ST | RB0 can also be the Capture1 input pin. |
| RB1/CAP2 | 54 | 58 | 46 | 70 | 56 | I/O | ST | RB1 can also be the Capture2 input pin. |
| RB2/PWM1 | 50 | 54 | 42 | 66 | 52 | I/O | ST | RB2 can also be the PWM1 output pin. |
| RB3/PWM2 | 53 | 57 | 45 | 69 | 55 | I/O | ST | RB3 can also be the PWM2 output pin. |
| RB4/TCLK12 | 52 | 56 | 44 | 68 | 54 | I/O | ST | RB4 can also be the external clock input to Timer1 and Timer2. |
| RB5/TCLK3 | 51 | 55 | 43 | 67 | 53 | I/O | ST | RB5 can also be the external clock input to Timer3. |
| RB6/SCK | 44 | 47 | 36 | 59 | 46 | I/O | ST | RB6 can also be used as the master/slave clock for the SPI. |
| RB7/SDO | 45 | 48 | 37 | 60 | 47 | I/O | ST | RB7 can also be used as the data output for the SPI. |

Legend: I = Input only; O = Output only; I/O = Input/Output;
P = Power; — = Not Used; TTL = TTL input; ST = Schmitt Trigger input

**Note 1:** The output is only available by the peripheral operation.
**2:** Open drain input/output pin. Pin forced to input upon any device RESET.

## 8.2 Table Writes to External Memory

Table writes to external memory are always two-cycle instructions. The second cycle writes the data to the external memory location. The sequence of events for an external memory write are the same for an internal write.

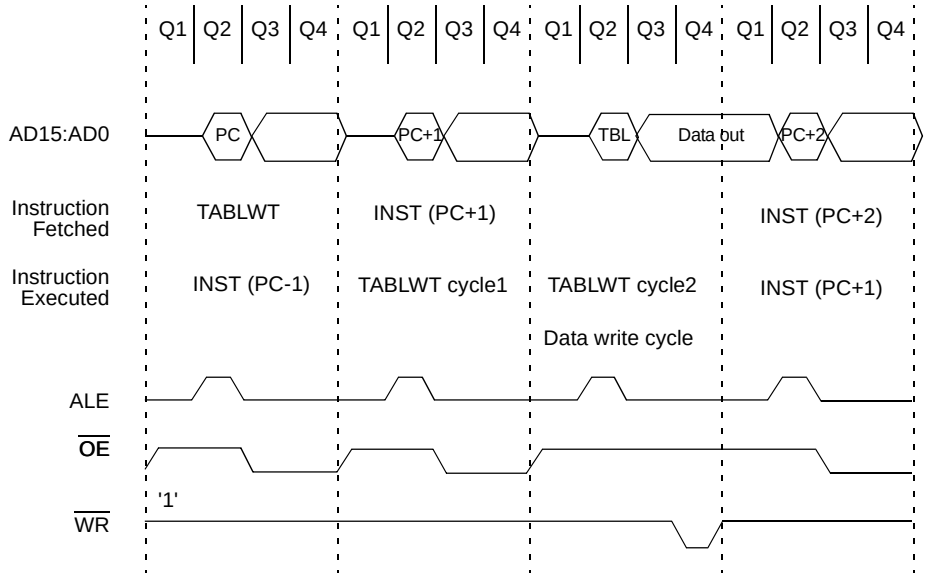
### 8.2.1 TABLE WRITE CODE

The "i" operand of the TABLWT instruction can specify that the value in the 16-bit TBLPTR register is automatically incremented (for the next write). In Example 8-1, the TBLPTR register is not automatically incremented.

**EXAMPLE 8-1: TABLE WRITE**

```
CLRWDT                  ; Clear WDT
MOVLW   HIGH (TBL_ADDR) ; Load the Table
MOVWF   TBLPTRH         ;   address
MOVLW   LOW (TBL_ADDR)  ;
MOVWF   TBLPTRL         ;
MOVLW   HIGH (DATA)     ; Load HI byte
TLWT    1, WREG         ;   in TABLATH
MOVLW   LOW (DATA)      ; Load LO byte
TABLWT  0,0,WREG        ;   in TABLATL
                        ;   and write to
                        ;   program memory
                        ;   (Ext. SRAM)
```

**FIGURE 8-5: TABLWT WRITE TIMING (EXTERNAL MEMORY)**




**Note:** If external write and GLINTD = '1' and Enable bit = '1', then when '1' → Flag bit, do table write. The highest pending interrupt is cleared.

## 10.2 PORTB and DDRB Registers

PORTB is an 8-bit wide, bi-directional port. The corresponding data direction register is DDRB. A '1' in DDRB configures the corresponding port pin as an input. A '0' in the DDRB register configures the corresponding port pin as an output. Reading PORTB reads the status of the pins, whereas writing to PORTB will write to the port latch.

Each of the PORTB pins has a weak internal pull-up. A single control bit can turn on all the pull-ups. This is done by clearing the $\overline{\text{RBPU}}$ (PORTA<7>) bit. The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are enabled on any RESET.

PORTB also has an interrupt-on-change feature. Only pins configured as inputs can cause this interrupt to occur (i.e., any RB7:RB0 pin configured as an output is excluded from the interrupt-on-change comparison). The input pins (of RB7:RB0) are compared with the value in the PORTB data latch. The “mismatch” outputs of RB7:RB0 are OR’d together to set the PORTB Interrupt Flag bit, RBIF (PIR1<7>).

This interrupt can wake the device from SLEEP. The user, in the Interrupt Service Routine, can clear the interrupt by:

a) Read-Write PORTB (such as: `MOVPF PORTB, PORTB`). This will end the mismatch condition.

b) Then, clear the RBIF bit.

A mismatch condition will continue to set the RBIF bit. Reading, then writing PORTB, will end the mismatch condition and allow the RBIF bit to be cleared.

This interrupt-on-mismatch feature, together with software configurable pull-ups on this port, allows easy interface to a keypad and makes it possible for wake-up on key depression. For an example, refer to Application Note AN552, “Implementing Wake-up on Keystroke.”

The interrupt-on-change feature is recommended for wake-up on operations, where PORTB is only used for the interrupt-on-change feature and key depression operations.

| **Note:** | On a device RESET, the RBIF bit is indeterminate, since the value in the latch may be different than the pin. |
|---|---|

**FIGURE 10-5: BLOCK DIAGRAM OF RB5:RB4 AND RB1:RB0 PORT PINS**

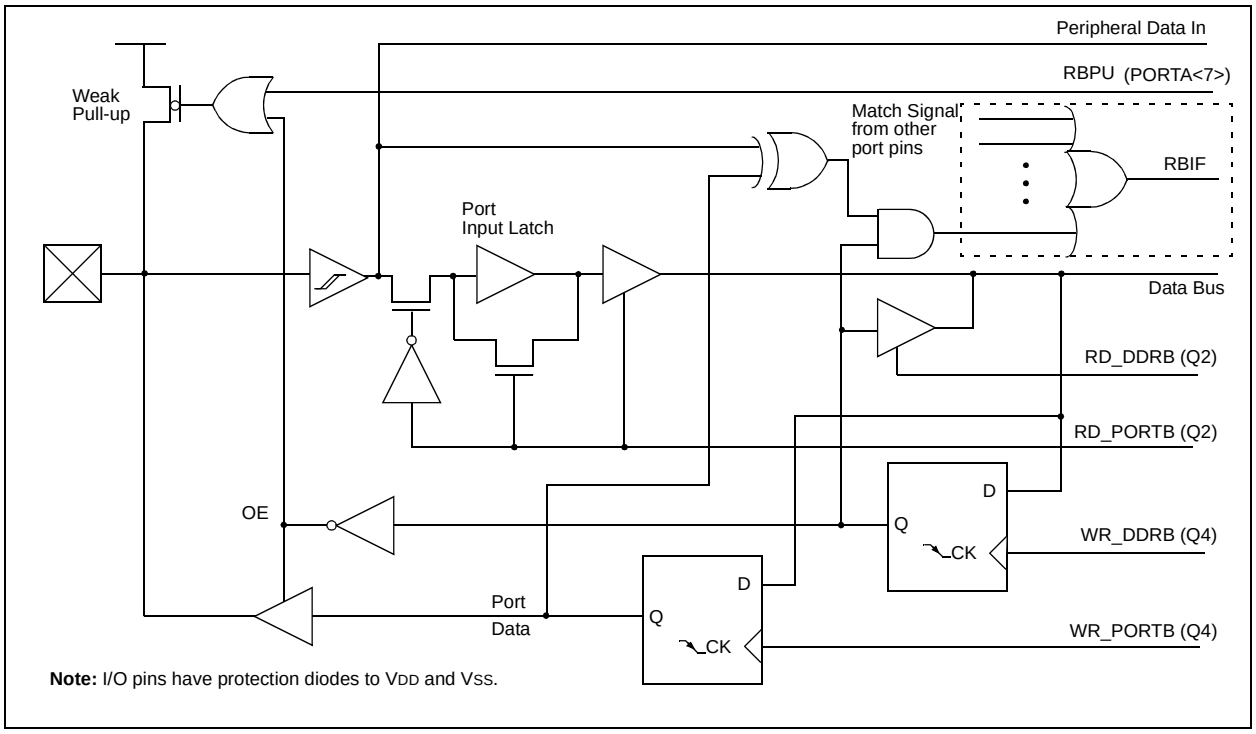
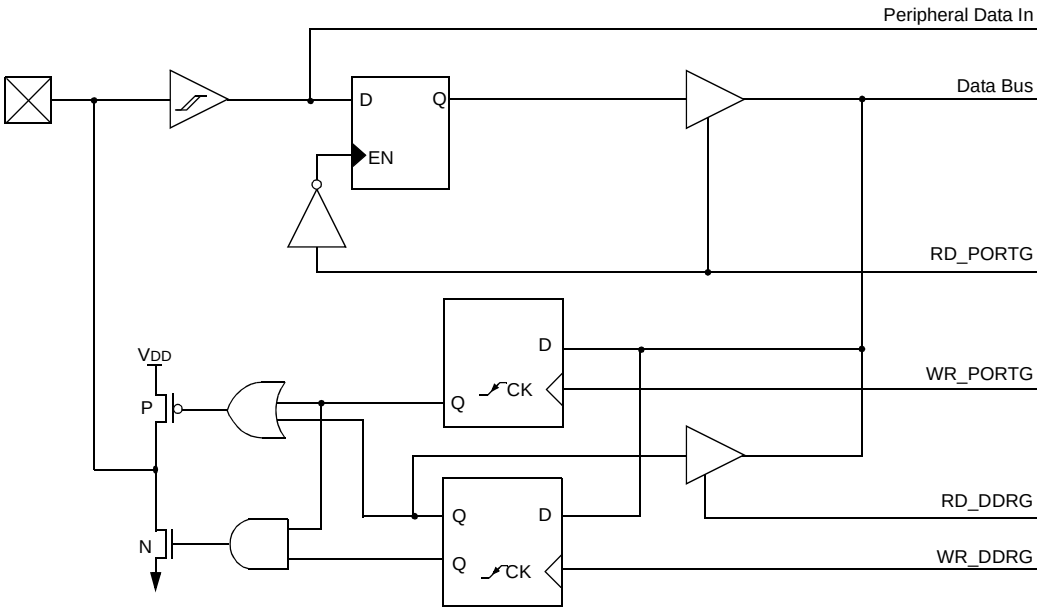
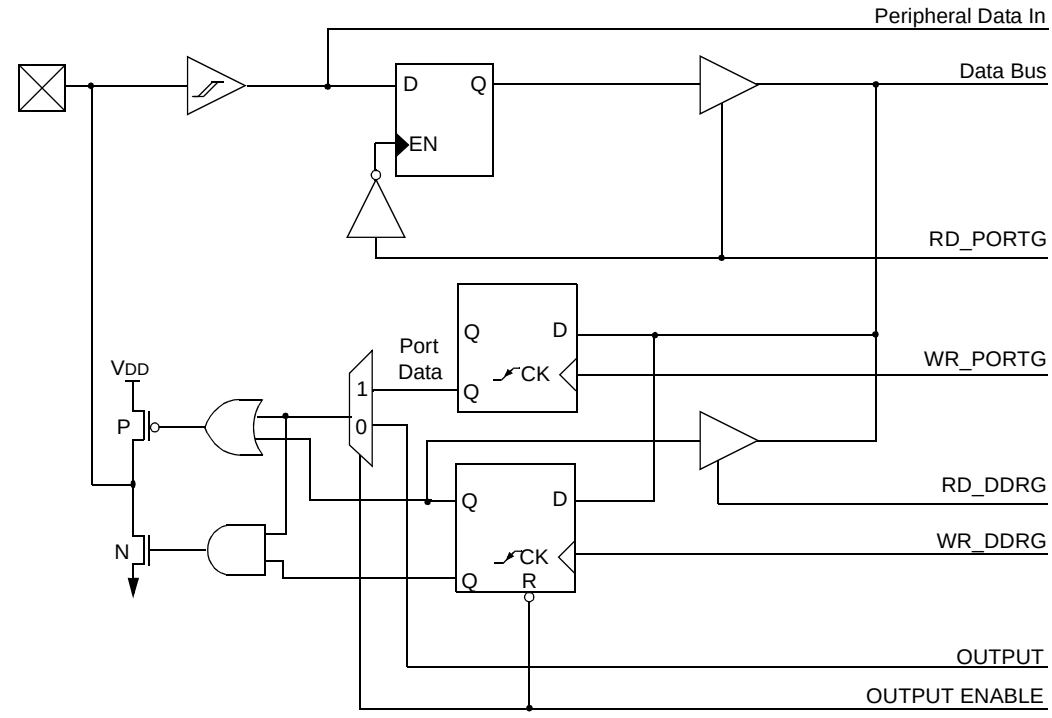


**FIGURE 10-15: RG4 BLOCK DIAGRAM**




**Note:** I/O pins have protection diodes to VDD and VSS.

**FIGURE 10-16: RG7:RG5 BLOCK DIAGRAM**




**Note:** I/O pins have protection diodes to VDD and VSS.

## 11.0 OVERVIEW OF TIMER RESOURCES

The PIC17C7XX has four timer modules. Each module can generate an interrupt to indicate that an event has occurred. These timers are called:

- Timer0 - 16-bit timer with programmable 8-bit prescaler
- Timer1 - 8-bit timer
- Timer2 - 8-bit timer
- Timer3 - 16-bit timer

For enhanced time base functionality, four input Captures and three Pulse Width Modulation (PWM) outputs are possible. The PWMs use the Timer1 and Timer2 resources and the input Captures use the Timer3 resource.

### 11.1 Timer0 Overview

The Timer0 module is a simple 16-bit overflow counter. The clock source can be either the internal system clock (Fosc/4) or an external clock.

When Timer0 uses an external clock source, it has the flexibility to allow user selection of the incrementing edge, rising or falling.

The Timer0 module also has a programmable prescaler. The T0PS3:T0PS0 bits (T0STA<4:1>) determine the prescale value. TMR0 can increment at the following rates: 1:1, 1:2, 1:4, 1:8, 1:16, 1:32, 1:64, 1:128, 1:256.

Synchronization of the external clock occurs after the prescaler. When the prescaler is used, the external clock frequency may be higher than the device's frequency. The maximum external frequency on the T0CKI pin is 50 MHz, given the high and low time requirements of the clock.

### 11.2 Timer1 Overview

The Timer1 module is an 8-bit timer/counter with an 8-bit period register (PR1). When the TMR1 value rolls over from the period match value to 0h, the TMR1IF flag is set and an interrupt will be generated if enabled. In Counter mode, the clock comes from the RB4/TCLK12 pin, which can also be selected to be the clock for the Timer2 module.

TMR1 can be concatenated with TMR2 to form a 16-bit timer. The TMR1 register is the LSB and TMR2 is the MSB. When in the 16-bit timer mode, there is a corresponding 16-bit period register (PR2:PR1). When the TMR2:TMR1 value rolls over from the period match value to 0h, the TMR1IF flag is set and an interrupt will be generated, if enabled.

### 11.3 Timer2 Overview

The Timer2 module is an 8-bit timer/counter with an 8-bit period register (PR2). When the TMR2 value rolls over from the period match value to 0h, the TMR2IF flag is set and an interrupt will be generated, if enabled. In Counter mode, the clock comes from the RB4/TCLK12 pin, which can also provide the clock for the Timer1 module.

TMR2 can be concatenated with TMR1 to form a 16-bit timer. The TMR2 register is the MSB and TMR1 is the LSB. When in the 16-bit timer mode, there is a corresponding 16-bit period register (PR2:PR1). When the TMR2:TMR1 value rolls over from the period match value to 0h, the TMR1IF flag is set and an interrupt will be generated, if enabled.

### 11.4 Timer3 Overview

The Timer3 module is a 16-bit timer/counter with a 16-bit period register. When the TMR3H:TMR3L value rolls over to 0h, the TMR3IF bit is set and an interrupt will be generated, if enabled. In Counter mode, the clock comes from the RB5/TCLK3 pin.

When operating in the four Capture modes, the period registers become the second (of four) 16-bit capture registers.

### 11.5 Role of the Timer/Counters

The timer modules are general purpose, but have dedicated resources associated with them. TImer1 and Timer2 are the time bases for the three Pulse Width Modulation (PWM) outputs, while Timer3 is the time base for the four input captures.

### 14.2.2 USART ASYNCHRONOUS RECEIVER

The receiver block diagram is shown in Figure 14-2. The data comes in the RX/DT pin and drives the data recovery block. The data recovery block is actually a high speed shifter operating at 16 times the baud rate, whereas the main receive serial shifter operates at the bit rate or at FOSC.

Once Asynchronous mode is selected, reception is enabled by setting bit CREN (RCSTA<4>).

The heart of the receiver is the receive (serial) shift register (RSR). After sampling the STOP bit, the received data in the RSR is transferred to the RCREG (if it is empty). If the transfer is complete, the interrupt bit, RCIF, is set. The actual interrupt can be enabled/disabled by setting/clearing the RCIE bit. RCIF is a read only bit which is cleared by the hardware. It is cleared when RCREG has been read and is empty. RCREG is a double buffered register (i.e., it is a two-deep FIFO). It is possible for two bytes of data to be received and transferred to the RCREG FIFO and a third byte begin shifting to the RSR. On detection of the STOP bit of the third byte, if the RCREG is still full, then the overrun error bit, OERR (RCSTA<1>) will be set. The word in the RSR will be lost. RCREG can be read twice to retrieve the two bytes in the FIFO. The OERR bit has to be cleared in software which is done by resetting the receive logic (CREN is set). If the OERR bit is set, transfers from the RSR to RCREG are inhibited, so it is essential to clear the OERR bit if it is set. The framing error bit FERR (RCSTA<2>) is set if a STOP bit is not detected.

| **Note:** | The FERR and the 9th receive bit are buffered the same way as the receive data. Reading the RCREG register will allow the RX9D and FERR bits to be loaded with values for the next received data. Therefore, it is essential for the user to read the RCSTA register before reading RCREG, in order not to lose the old FERR and RX9D information. |
|---|---|

### 14.2.3 SAMPLING

The data on the RX/DT pin is sampled three times by a majority detect circuit to determine if a high or a low level is present at the RX/DT pin. The sampling is done on the seventh, eighth and ninth falling edges of a x16 clock (Figure 14-5).

The x16 clock is a free running clock and the three sample points occur at a frequency of every 16 falling edges.

**FIGURE 14-5: RX PIN SAMPLING SCHEME**

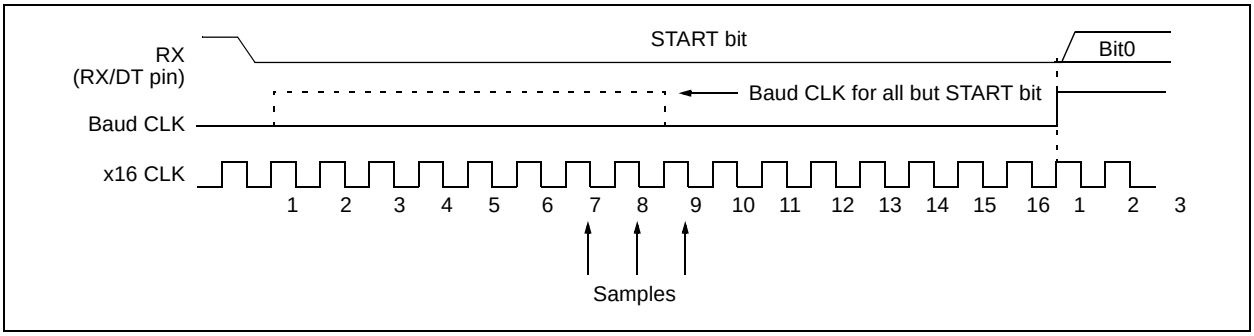
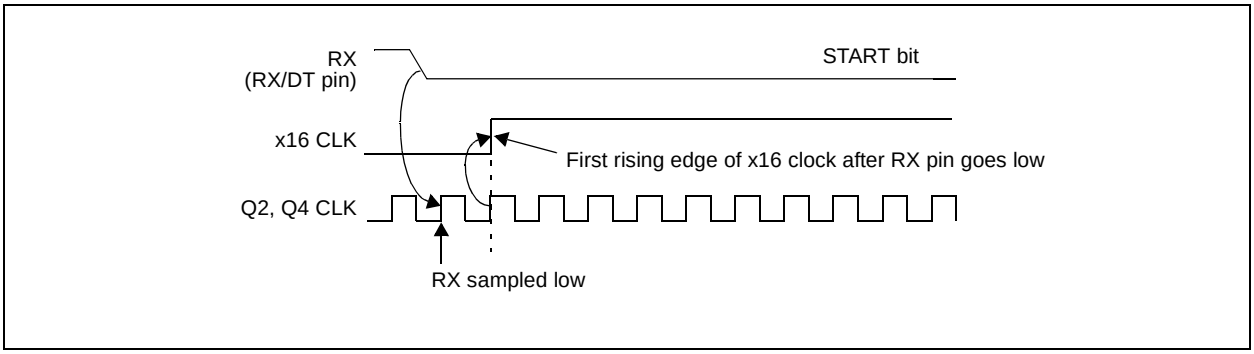



**FIGURE 14-6: START BIT DETECT**

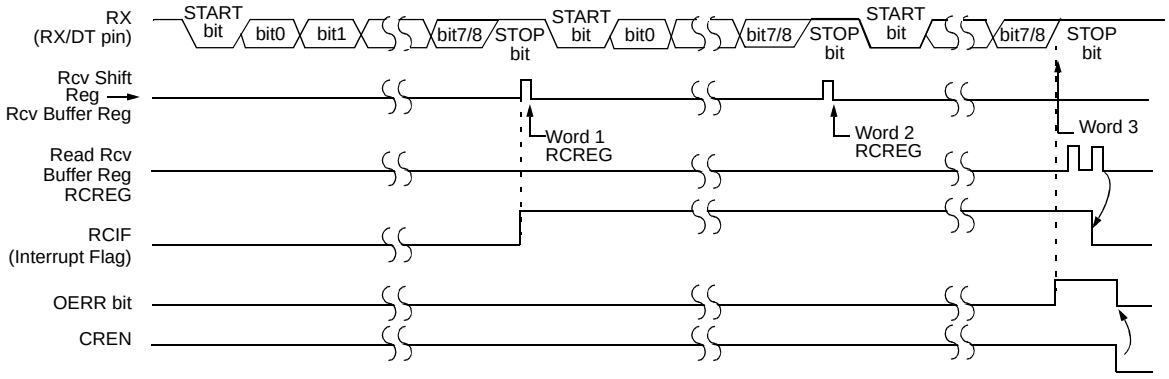


Steps to follow when setting up an Asynchronous Reception:

1. Initialize the SPBRG register for the appropriate baud rate.
2. Enable the asynchronous serial port by clearing the SYNC bit and setting the SPEN bit.
3. If interrupts are desired, then set the RCIE bit.
4. If 9-bit reception is desired, then set the RX9 bit.
5. Enable the reception by setting the CREN bit.
6. The RCIF bit will be set when reception completes and an interrupt will be generated if the RCIE bit was set.
7. Read RCSTA to get the ninth bit (if enabled) and FERR bit to determine if any error occurred during reception.
8. Read RCREG for the 8-bit received data.
9. If an overrun error occurred, clear the error by clearing the OERR bit.

**Note:** To terminate a reception, either clear the SREN and CREN bits, or the SPEN bit. This will reset the receive logic, so that it will be in the proper state when receive is re-enabled.

**FIGURE 14-7: ASYNCHRONOUS RECEPTION**




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

**TABLE 14-7: REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION**

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | $\overline{\text{MCLR}}$, WDT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 16h, Bank 1 | PIR1 | RBIF | TMR3IF | TMR2IF | TMR1IF | CA2IF | CA1IF | TX1IF | RC1IF | x000 0010 | u000 0010 |
| 17h, Bank 1 | PIE1 | RBIE | TMR3IE | TMR2IE | TMR1IE | CA2IE | CA1IE | TX1IE | RC1IE | 0000 0000 | 0000 0000 |
| 13h, Bank 0 | RCSTA1 | SPEN | RX9 | SREN | CREN | — | FERR | OERR | RX9D | 0000 -00x | 0000 -00u |
| 14h, Bank 0 | RCREG1 | RX7 | RX6 | RX5 | RX4 | RX3 | RX2 | RX1 | RX0 | xxxx xxxx | uuuu uuuu |
| 15h, Bank 0 | TXSTA1 | CSRC | TX9 | TXEN | SYNC | — | — | TRMT | TX9D | 0000 --1x | 0000 --1u |
| 17h, Bank 0 | SPBRG1 | Baud Rate Generator Register | | | | | | | | 0000 0000 | 0000 0000 |
| 10h, Bank 4 | PIR2 | SSPIF | BCLIF | ADIF | — | CA4IF | CA3IF | TX2IF | RC2IF | 000- 0010 | 000- 0010 |
| 11h, Bank 4 | PIE2 | SSPIE | BCLIE | ADIE | — | CA4IE | CA3IE | TX2IE | RC2IE | 000- 0000 | 000- 0000 |
| 13h, Bank 4 | RCSTA2 | SPEN | RX9 | SREN | CREN | — | FERR | OERR | RX9D | 0000 -00x | 0000 -00u |
| 14h, Bank 4 | RCREG2 | RX7 | RX6 | RX5 | RX4 | RX3 | RX2 | RX1 | RX0 | xxxx xxxx | uuuu uuuu |
| 15h, Bank 4 | TXSTA2 | CSRC | TX9 | TXEN | SYNC | — | — | TRMT | TX9D | 0000 --1x | 0000 --1u |
| 17h, Bank 4 | SPBRG2 | Baud Rate Generator Register | | | | | | | | 0000 0000 | 0000 0000 |

Legend: x = unknown, u = unchanged, - = unimplemented, read as a '0'. Shaded cells are not used for asynchronous reception.

The SSPSTAT register gives the status of the data transfer. This information includes detection of a START or STOP bit, specifies if the received byte was data or address if the next byte is the completion of 10-bit address and if this will be a read or write data transfer.

The SSPBUF is the register to which transfer data is written to or read from. The SSPSR register shifts the data in or out of the device. In receive operations, the SSPBUF and SSPSR create a doubled buffered receiver. This allows reception of the next byte to begin before reading the last byte of received data. When the complete byte is received, it is transferred to the SSPBUF register and flag bit SSPIF is set. If another complete byte is received before the SSPBUF register is read, a receiver overflow has occurred and bit SSPOV (SSPCON1<6>) is set and the byte in the SSPSR is lost.

The SSPADD register holds the slave address. In 10-bit mode, the user needs to write the high byte of the address (1111 0 A9 A8 0). Following the high byte address match, the low byte of the address needs to be loaded (A7:A0).

### 15.2.1 SLAVE MODE

In Slave mode, the SCL and SDA pins must be configured as inputs. The MSSP module will override the input state with the output data when required (slave-transmitter).

When an address is matched or the data transfer after an address match is received, the hardware automatically will generate the acknowledge ($\overline{ACK}$) pulse and then load the SSPBUF register with the received value currently in the SSPSR register.

There are certain conditions that will cause the MSSP module not to give this $\overline{ACK}$ pulse. These are if either (or both):

a) The buffer full bit BF (SSPSTAT<0>) was set before the transfer was received.

b) The overflow bit SSPOV (SSPCON1<6>) was set before the transfer was received.

If the BF bit is set, the SSPSR register value is not loaded into the SSPBUF, but bit SSPIF and SSPOV are set. Table 15-2 shows what happens when a data transfer byte is received, given the status of bits BF and SSPOV. The shaded cells show the condition where user software did not properly clear the overflow condition. Flag bit BF is cleared by reading the SSPBUF register, while bit SSPOV is cleared through software.

The SCL clock input must have a minimum high and low time for proper operation. The high and low times of the $I^2C$ specification, as well as the requirement of the MSSP module, are shown in timing parameter #100 and parameter #101 of the Electrical Specifications.

### 15.2.6 MULTI-MASTER MODE

In Multi-Master mode, the interrupt generation on the detection of the START and STOP conditions allows the determination of when the bus is free. The STOP (P) and START (S) bits are cleared from a RESET, or when the MSSP module is disabled. Control of the $I^2C$ bus may be taken when bit P (SSPSTAT<4>) is set, or the bus is idle, with both the S and P bits clear. When the bus is busy, enabling the SSP interrupt will generate the interrupt when the STOP condition occurs.

In Multi-Master operation, the SDA line must be monitored for arbitration, to see if the signal level is the expected output level. This check is performed in hardware, with the result placed in the BCLIF bit.

The states where arbitration can be lost are:

- Address Transfer
- Data Transfer
- A START Condition
- A Repeated Start Condition
- An Acknowledge Condition

### 15.2.7 $I^2C$ MASTER MODE SUPPORT

Master mode is enabled by setting and clearing the appropriate SSPM bits in SSPCON1 and by setting the SSPEN bit. Once Master mode is enabled, the user has six options.

- Assert a START condition on SDA and SCL.
- Assert a Repeated Start condition on SDA and SCL.
- Write to the SSPBUF register initiating transmission of data/address.
- Generate a STOP condition on SDA and SCL.
- Configure the $I^2C$ port to receive data.
- Generate an Acknowledge condition at the end of a received byte of data.

| | |
|---|---|
| **Note:** | The MSSP Module, when configured in $I^2C$ Master mode, does not allow queueing of events. For instance: The user is not allowed to initiate a START condition and immediately write the SSPBUF register to initiate transmission before the START condition is complete. In this case, the SSPBUF will not be written to and the WCOL bit will be set, indicating that a write to the SSPBUF did not occur. |

#### 15.2.7.1 $I^2C$ Master Mode Operation

The master device generates all of the serial clock pulses and the START and STOP conditions. A transfer is ended with a STOP condition or with a Repeated Start condition. Since the Repeated Start condition is also the beginning of the next serial transfer, the $I^2C$ bus will not be released.

In Master Transmitter mode, serial data is output through SDA, while SCL outputs the serial clock. The first byte transmitted contains the slave address of the receiving device (7 bits) and the Read/Write (R/$\overline{W}$) bit. In this case, the R/$\overline{W}$ bit will be logic '0'. Serial data is transmitted 8 bits at a time. After each byte is transmitted, an acknowledge bit is received. START and STOP conditions are output to indicate the beginning and the end of a serial transfer.

In Master Receive mode, the first byte transmitted contains the slave address of the transmitting device (7 bits) and the R/$\overline{W}$ bit. In this case, the R/$\overline{W}$ bit will be logic '1'. Thus, the first byte transmitted is a 7-bit slave address, followed by a '1' to indicate receive bit. Serial data is received via SDA, while SCL outputs the serial clock. Serial data is received 8 bits at a time. After each byte is received, an acknowledge bit is transmitted. START and STOP conditions indicate the beginning and end of transmission.

The baud rate generator used for SPI mode operation is now used to set the SCL clock frequency for either 100 kHz, 400 kHz, or 1 MHz $I^2C$ operation. The baud rate generator reload value is contained in the lower 7 bits of the SSPADD register. The baud rate generator will automatically begin counting on a write to the SSPBUF. Once the given operation is complete (i.e., transmission of the last data bit is followed by ACK), the internal clock will automatically stop counting and the SCL pin will remain in its last state

**BSF** **Bit Set f**

Syntax: [ *label* ] BSF f,b

Operands: $0 \le f \le 255$
$0 \le b \le 7$

Operation: $1 \rightarrow (f<b>)$

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1000 | 0bbb | ffff | ffff |

Description: Bit 'b' in register 'f' is set.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write register 'f' |

Example: `BSF FLAG_REG, 7`

Before Instruction
FLAG_REG = 0x0A

After Instruction
FLAG_REG = 0x8A

**BTFSC** **Bit Test, skip if Clear**

Syntax: [ *label* ] BTFSC f,b

Operands: $0 \le f \le 255$
$0 \le b \le 7$

Operation: skip if (f<b>) = 0

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1001 | 1bbb | ffff | ffff |

Description: If bit 'b' in register 'f' is 0, then the next instruction is skipped.

If bit 'b' is 0, then the next instruction fetched during the current instruction execution is discarded and a `NOP` is executed instead, making this a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | No operation |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

Example:
```
HERE    BTFSC   FLAG,1
FALSE   :
TRUE    :
```

Before Instruction
PC = address `(HERE)`

After Instruction
If FLAG<1> = 0;
PC = address `(TRUE)`
If FLAG<1> = 1;
PC = address `(FALSE)`

**DCFSNZ** **Decrement f, skip if not 0**

Syntax: [*label*] DCFSNZ f,d

Operands: $0 \le f \le 255$
$d \in [0,1]$

Operation: (f) – 1 → (dest);
skip if not 0

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0010 | 011d | ffff | ffff |

Description: The contents of register 'f' are decremented. If 'd' is 0, the result is placed in WREG. If 'd' is 1, the result is placed back in register 'f'.
If the result is not 0, the next instruction, which is already fetched is discarded and a NOP is executed instead, making it a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

Example:
```
HERE    DCFSNZ  TEMP, 1
ZERO    :
NZERO   :
```

Before Instruction
TEMP_VALUE = ?

After Instruction
TEMP_VALUE = TEMP_VALUE - 1,
If TEMP_VALUE = 0;
PC = Address (ZERO)
If TEMP_VALUE ≠ 0;
PC = Address (NZERO)

**GOTO** **Unconditional Branch**

Syntax: [ *label* ] GOTO k

Operands: $0 \le k \le 8191$

Operation: k → PC<12:0>;
k<12:8> → PCLATH<4:0>,
PC<15:13> → PCLATH<7:5>

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 110k | kkkk | kkkk | kkkk |

Description: GOTO allows an unconditional branch anywhere within an 8K page boundary. The thirteen-bit immediate value is loaded into PC bits <12:0>. Then the upper eight bits of PC are loaded into PCLATH. GOTO is always a two-cycle instruction.

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

Example: GOTO THERE

After Instruction
PC = Address (THERE)

**RETURN** **Return from Subroutine**

Syntax: [ *label* ] RETURN

Operands: None

Operation: TOS → PC;

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0000 | 0010 |

Description: Return from subroutine. The stack is popped and the top of the stack (TOS) is loaded into the program counter.

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | Process Data | POP PC from stack |
| No operation | No operation | No operation | No operation |

Example: `RETURN`

After Interrupt
PC = TOS

**RLCF** **Rotate Left f through Carry**

Syntax: [ *label* ] RLCF f,d

Operands: $0 \le f \le 255$
$d \in [0,1]$

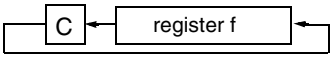
Operation: f<n> → d<n+1>;
f<7> → C;
C → d<0>

Status Affected: C

Encoding:

| | | | |
|---|---|---|---|
| 0001 | 101d | ffff | ffff |

Description: The contents of register 'f' are rotated one bit to the left through the Carry Flag. If 'd' is 0, the result is placed in WREG. If 'd' is 1, the result is stored back in register 'f'.




Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: `RLCF REG,0`

Before Instruction
REG = 1110 0110
C = 0

After Instruction
REG = 1110 0110
WREG = 1100 1100
C = 1

**TLWT** **Table Latch Write**

Syntax: [ *label* ] TLWT t,f

Operands: $0 \le f \le 255$
$t \in [0,1]$

Operation: If t = 0,
f → TBLATL;
If t = 1,
f → TBLATH

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1010 | 01tx | ffff | ffff |

Description: Data from file register 'f' is written into the 16-bit table latch (TBLAT).
If t = 1; high byte is written
If t = 0; low byte is written
This instruction is used in conjunction with TABLWT to transfer data from data memory to program memory.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write register TBLATH or TBLATL |

Example: TLWT t, RAM

Before Instruction
t = 0
RAM = 0xB7
TBLAT = 0x0000 (TBLATH = 0x00) (TBLATL = 0x00)

After Instruction
RAM = 0xB7
TBLAT = 0x00B7 (TBLATH = 0x00) (TBLATL = 0xB7)

Before Instruction
t = 1
RAM = 0xB7
TBLAT = 0x0000 (TBLATH = 0x00) (TBLATL = 0x00)

After Instruction
RAM = 0xB7
TBLAT = 0xB700 (TBLATH = 0xB7) (TBLATL = 0x00)

**TSTFSZ** **Test f, skip if 0**

Syntax: [ *label* ] TSTFSZ f

Operands: $0 \le f \le 255$

Operation: skip if f = 0

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0011 | 0011 | ffff | ffff |

Description: If 'f' = 0, the next instruction, fetched during the current instruction execution, is discarded and a NOP is executed, making this a two-cycle instruction.

Words: 1

Cycles: 1 (2)

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | No operation |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

Example:
```
HERE    TSTFSZ  CNT
NZERO     :
ZERO      :
```

Before Instruction
PC = Address (HERE)

After Instruction
If CNT = 0x00,
PC = Address (ZERO)
If CNT ¼ 0x00,
PC = Address (NZERO)

**XORLW** **Exclusive OR Literal with WREG**

Syntax: [ *label* ] XORLW k

Operands: $0 \le k \le 255$

Operation: (WREG) .XOR. k → (WREG)

Status Affected: Z

Encoding:

| | | | |
|---|---|---|---|
| 1011 | 0100 | kkkk | kkkk |

Description: The contents of WREG are XOR'ed with the 8-bit literal 'k'. The result is placed in WREG.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write to WREG |

Example: `XORLW 0xAF`

Before Instruction
WREG = 0xB5

After Instruction
WREG = 0x1A

**XORWF** **Exclusive OR WREG with f**

Syntax: [ *label* ] XORWF f,d

Operands: $0 \le f \le 255$
$d \in [0,1]$

Operation: (WREG) .XOR. (f) → (dest)

Status Affected: Z

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 110d | ffff | ffff |

Description: Exclusive OR the contents of WREG with register 'f'. If 'd' is 0, the result is stored in WREG. If 'd' is 1, the result is stored back in the register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: `XORWF REG, 1`

Before Instruction
REG = 0xAF 1010 1111
WREG = 0xB5 1011 0101

After Instruction
REG = 0x1A 0001 1010
WREG = 0xB5

| DC CHARACTERISTICS | | | Standard Operating Conditions (unless otherwise stated)<br>Operating temperature<br>-40°C ≤ TA ≤ +125°C for extended<br>-40°C ≤ TA ≤ +85°C for industrial<br>0°C ≤ TA ≤ +70°C for commercial<br>Operating voltage VDD range as described in Section 20.1 | | | | |
|---|---|---|---|---|---|---|---|
| **Param. No.** | **Sym** | **Characteristic** | **Min** | **Typ†** | **Max** | **Units** | **Conditions** |
| D150 | VOD | **Open Drain High Voltage** | – | – | 8.5 | V | RA2 and RA3 pins only pulled up to externally applied voltage |
| | | **Capacitive Loading Specs on Output Pins** | | | | | |
| D100 | COSC2 | OSC2/CLKOUT pin | – | – | 25 | pF | In EC or RC osc modes, when OSC2 pin is outputting CLKOUT. External clock is used to drive OSC1. |
| D101 | CIO | All I/O pins and OSC2 (in RC mode) | – | – | 50 | pF | |
| D102 | CAD | System Interface Bus (PORTC, PORTD and PORTE) | – | – | 50 | pF | In Microprocessor or Extended Microcontroller mode |
| | | **Internal Program Memory Programming Specs (Note 4)** | | | | | |
| D110 | VPP | Voltage on MCLR/VPP pin | 12.75 | – | 13.25 | V | **(Note 5)** |
| D111 | VDDP | Supply voltage during programming | 4.75 | 5.0 | 5.25 | V | |
| D112 | IPP | Current into MCLR/VPP pin | – | 25 | 50 | mA | |
| D113 | IDDP | Supply current during programming | – | – | 30 | mA | |
| D114 | TPROG | Programming pulse width | 100 | – | 1000 | ms | Terminated via internal/ external interrupt or a RESET |

† Data in "Typ" column is at 5V, 25°C unless otherwise stated.

**Note 1:** In RC oscillator configuration, the OSC1/CLKIN pin is a Schmitt Trigger input. It is not recommended that the PIC17CXXX devices be driven with external clock in RC mode.

**2:** The leakage current on the MCLR pin is strongly dependent on the applied voltage level. Higher leakage current may be measured at different input voltages.

**3:** Negative current is defined as current sourced by the pin.

**4:** These specifications are for the programming of the on-chip program memory EPROM through the use of the table write instructions. The complete programming specifications can be found in: PIC17C7XX Programming Specifications (Literature number DS TBD).

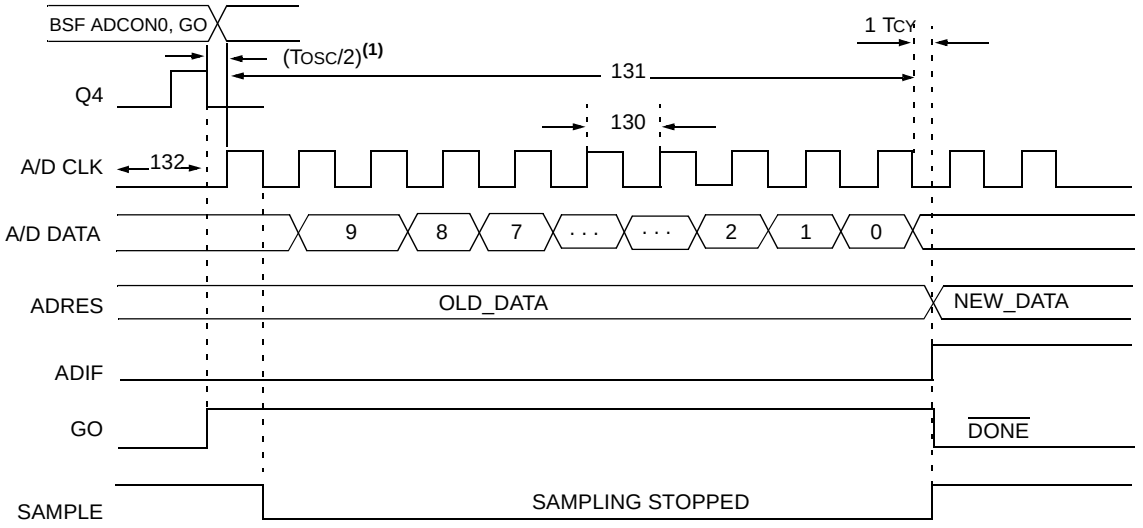
**5:** The MCLR/VPP pin may be kept in this range at times other than programming, but is not recommended.

**6:** For TTL buffers, the better of the two specifications may be used.

**Note 1:** When using the Table Write for internal programming, the device temperature must be less than 40°C.

**2:** For In-Circuit Serial Programming (ICSP™), refer to the device programming specification.

**FIGURE 20-23: A/D CONVERSION TIMING**




**Note 1:** If the A/D clock source is selected as RC, a time of TCY is added before the A/D clock starts. This allows the SLEEP instruction to be executed.

**TABLE 20-19: A/D CONVERSION REQUIREMENTS**

| Param. No. | Sym | Characteristic | | Min | Typ† | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|---|
| 130 | TAD | A/D clock period | PIC17**C**XXX | 1.6 | — | — | µs | TOSC based, VREF ≥ 3.0V |
| | | | PIC17**LC**XXX | 3.0 | — | — | µs | TOSC based, VREF full range |
| | | | PIC17**C**XXX | 2.0 | 4.0 | 6.0 | µs | A/D RC mode |
| | | | PIC17**LC**XXX | 3.0 | 6.0 | 9.0 | µs | A/D RC mode |
| 131 | TCNV | Conversion time (not including acquisition time) **(Note 1)** | | 11 | — | 12 | Tad | |
| 132 | TACQ | Acquisition time | | **(Note 2)** | 20 | — | µs | |
| | | | | 10 | — | — | µs | The minimum time is the amplifier settling time. This may be used if the "new" input voltage has not changed by more than 1LSb (i.e., 5 mV @ 5.12V) from the last sampled voltage (as stated on CHOLD). |
| 134 | TGO | Q4 to ADCLK start | | — | Tosc/2 | — | — | If the A/D clock source is selected as RC, a time of TCY is added before the A/D clock starts. This allows the SLEEP instruction to be executed. |

† Data in "Typ" column is at 5V, 25°C unless otherwise stated.

**Note 1:** ADRES register may be read on the following TCY cycle.

**2:** See Section 16.1 for minimum conditions when input voltage has changed more than 1 LSb.

**FIGURE 21-2: TYPICAL RC OSCILLATOR FREQUENCY vs. VDD**

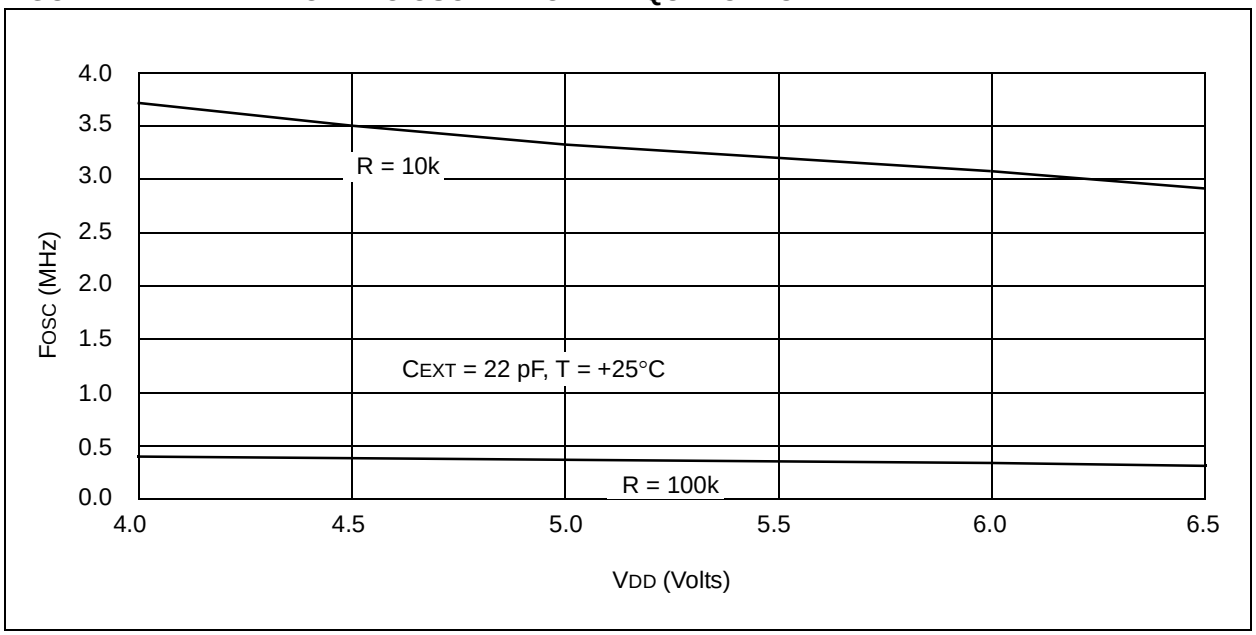



**FIGURE 21-3: TYPICAL RC OSCILLATOR FREQUENCY vs. VDD**

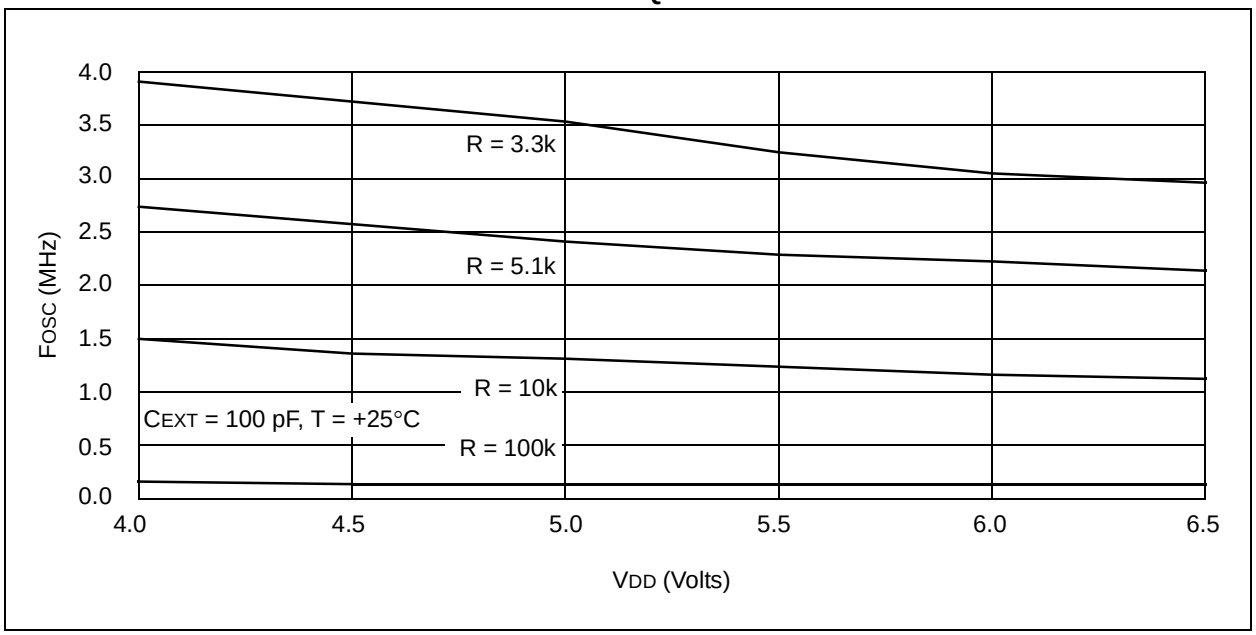


**FIGURE 21-23: MAXIMUM AND MINIMUM VIN vs. VDD ($I^2C$ Input, -40°C to +125°C)**