**Welcome to E-XFL.COM**

## What is "Embedded - Microcontrollers"?

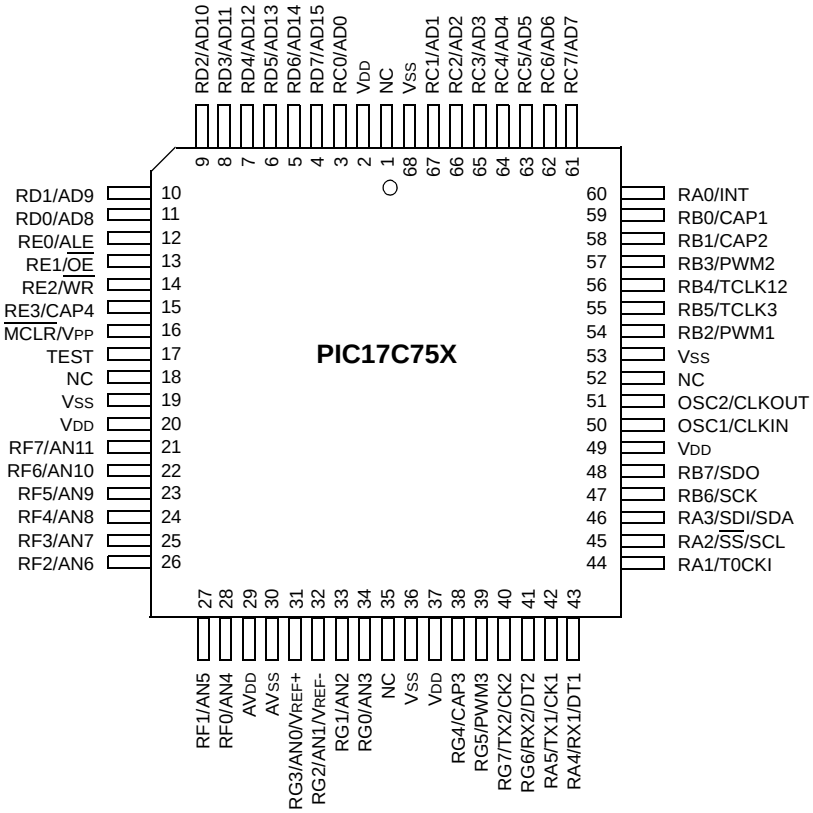
"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" 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"

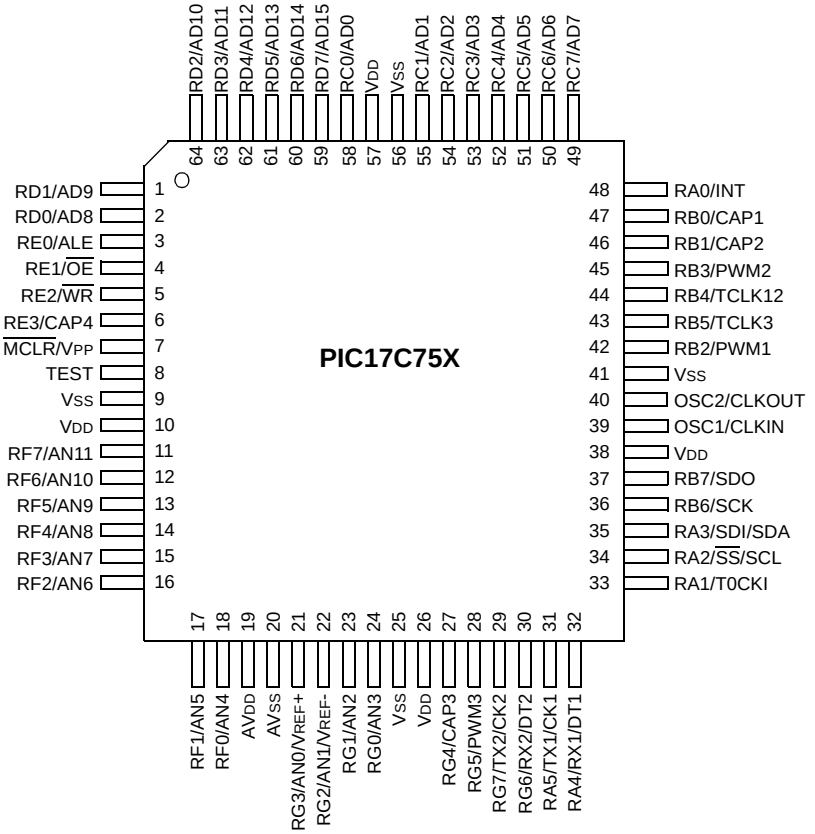
| Details | |
|---|---|
| Product Status | Active |
| Core Processor | PIC |
| Core Size | 8-Bit |
| Speed | 8MHz |
| Connectivity | I²C, SPI, UART/USART |
| Peripherals | Brown-out Detect/Reset, POR, PWM, WDT |
| Number of I/O | 66 |
| Program Memory Size | 32KB (16K x 16) |
| Program Memory Type | OTP |
| EEPROM Size | - |
| RAM Size | 902 x 8 |
| Voltage - Supply (Vcc/Vdd) | 3V ~ 5.5V |
| Data Converters | A/D 16x10b |
| Oscillator Type | External |
| Operating Temperature | -40°C ~ 85°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/pic17lc766-08i-l |

## Pin Diagrams cont.'d

### 68-Pin PLCC



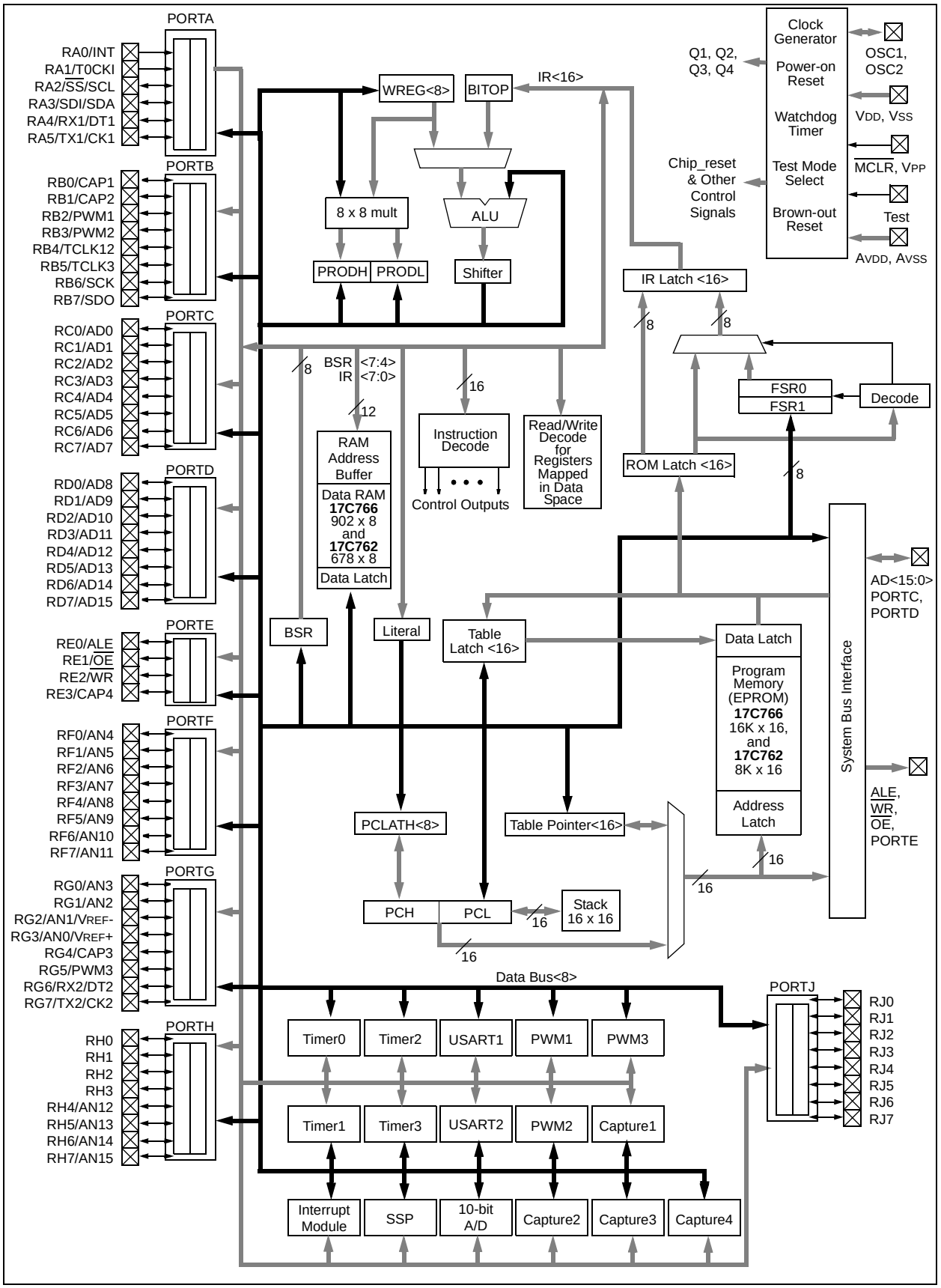

### 64-Pin TQFP





© 1998-2013 Microchip Technology Inc.

**FIGURE 3-2: PIC17C762/766 BLOCK DIAGRAM**



### 5.1.4 TIME-OUT SEQUENCE

On power-up, the time-out sequence is as follows: First, the internal POR signal goes high when the POR trip point is reached. If MCLR is high, then both the OST and PWRT timers start. In general, the PWRT time-out is longer, except with low frequency crystals/resonators. The total time-out also varies based on oscillator configuration. Table 5-1 shows the times that are associated with the oscillator configuration. Figure 5-5 and Figure 5-6 display these time-out sequences.

If the device voltage is not within electrical specification at the end of a time-out, the MCLR/VPP pin must be held low until the voltage is within the device specification. The use of an external RC delay is sufficient for many of these applications.

The time-out sequence begins from the first rising edge of MCLR.

Table 5-3 shows the RESET conditions for some special registers, while Table 5-4 shows the initialization conditions for all the registers.

**TABLE 5-1: TIME-OUT IN VARIOUS SITUATIONS**

| Oscillator Configuration | POR, BOR | Wake-up from SLEEP | MCLR Reset |
|---|---|---|---|
| XT, LF | Greater of: 96 ms or 1024TOSC | 1024TOSC | — |
| EC, RC | Greater of: 96 ms or 1024TOSC | — | — |

**TABLE 5-2: STATUS BITS AND THEIR SIGNIFICANCE**

| POR | BOR(1) | TO | PD | Event |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | Power-on Reset |
| 1 | 1 | 1 | 0 | MCLR Reset during SLEEP or interrupt wake-up from SLEEP |
| 1 | 1 | 0 | 1 | WDT Reset during normal operation |
| 1 | 1 | 0 | 0 | WDT Wake-up during SLEEP |
| 1 | 1 | 1 | 1 | MCLR Reset during normal operation |
| 1 | 0 | 1 | 1 | Brown-out Reset |
| 0 | 0 | 0 | x | Illegal, TO is set on POR |
| 0 | 0 | x | 0 | Illegal, PD is set on POR |
| x | x | 1 | 1 | CLRWDT instruction executed |

**Note 1:** When BODEN is enabled, else the BOR status bit is unknown.

**TABLE 5-3: RESET CONDITION FOR THE PROGRAM COUNTER AND THE CPUSTA REGISTER**

| Event | | PCH:PCL | CPUSTA(4) | OST Active |
|---|---|---|---|---|
| Power-on Reset | | 0000h | --11 1100 | Yes |
| Brown-out Reset | | 0000h | --11 1110 | Yes |
| MCLR Reset during normal operation | | 0000h | --11 1111 | No |
| MCLR Reset during SLEEP | | 0000h | --11 1011 | Yes(2) |
| WDT Reset during normal operation | | 0000h | --11 0111 | No |
| WDT Reset during SLEEP(3) | | 0000h | --11 0011 | Yes(2) |
| Interrupt Wake-up from SLEEP | GLINTD is set | PC + 1 | --11 1011 | Yes(2) |
| | GLINTD is clear | PC + 1(1) | --10 1011 | Yes(2) |

Legend: u = unchanged, x = unknown, - = unimplemented, read as '0'

**Note 1:** On wake-up, this instruction is executed. The instruction at the appropriate interrupt vector is fetched and then executed.

**2:** The OST is only active (on wake-up) when the oscillator is configured for XT or LF modes.

**3:** The Program Counter = 0; that is, the device branches to the RESET vector and places SFRs in WDT Reset states. This is different from the mid-range devices.

**4:** When BODEN is enabled, else the BOR status bit is unknown.

**TABLE 5-4: INITIALIZATION CONDITIONS FOR SPECIAL FUNCTION REGISTERS (CONTINUED)**

| Register | Address | Power-on Reset<br>Brown-out Reset | $\overline{\text{MCLR}}$ Reset<br>WDT Reset | Wake-up from SLEEP through Interrupt |
|---|---|---|---|---|
| **Bank 4** | | | | |
| PIR2 | 10h | 000- 0010 | 000- 0010 | uuu- uuuu(1) |
| PIE2 | 11h | 000- 0000 | 000- 0000 | uuu- uuuu |
| Unimplemented | 12h | ---- ---- | ---- ---- | ---- ---- |
| RCSTA2 | 13h | 0000 -00x | 0000 -00u | uuuu -uuu |
| RCREG2 | 14h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXSTA2 | 15h | 0000 --1x | 0000 --1u | uuuu --uu |
| TXREG2 | 16h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| SPBRG2 | 17h | 0000 0000 | 0000 0000 | uuuu uuuu |
| **Bank 5** | | | | |
| DDRF | 10h | 1111 1111 | 1111 1111 | uuuu uuuu |
| PORTF(4) | 11h | 0000 0000 | 0000 0000 | uuuu uuuu |
| DDRG | 12h | 1111 1111 | 1111 1111 | uuuu uuuu |
| PORTG(4) | 13h | xxxx 0000 | uuuu 0000 | uuuu uuuu |
| ADCON0 | 14h | 0000 -0-0 | 0000 -0-0 | uuuu uuuu |
| ADCON1 | 15h | 000- 0000 | 000- 0000 | uuuu uuuu |
| ADRESL | 16h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| ADRESH | 17h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| **Bank 6** | | | | |
| SSPADD | 10h | 0000 0000 | 0000 0000 | uuuu uuuu |
| SSPCON1 | 11h | 0000 0000 | 0000 0000 | uuuu uuuu |
| SSPCON2 | 12h | 0000 0000 | 0000 0000 | uuuu uuuu |
| SSPSTAT | 13h | 0000 0000 | 0000 0000 | uuuu uuuu |
| SSPBUF | 14h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| Unimplemented | 15h | ---- ---- | ---- ---- | ---- ---- |
| Unimplemented | 16h | ---- ---- | ---- ---- | ---- ---- |
| Unimplemented | 17h | ---- ---- | ---- ---- | ---- ---- |

Legend: u = unchanged, x = unknown, - = unimplemented, read as '0', q = value depends on condition

**Note 1:** One or more bits in INTSTA, PIR1, PIR2 will be affected (to cause wake-up).

**2:** When the wake-up is due to an interrupt and the GLINTD bit is cleared, the PC is loaded with the interrupt vector.

**3:** See Table 5-3 for RESET value of specific condition.

**4:** This is the value that will be in the port output latch.

**5:** When the device is configured for Microprocessor or Extended Microcontroller mode, the operation of this port does not rely on these registers.

**6:** On any device RESET, these pins are configured as inputs.

## 6.4 Interrupt Operation

Global Interrupt Disable bit, GLINTD (CPUSTA<4>), enables all unmasked interrupts (if clear), or disables all interrupts (if set). Individual interrupts can be disabled through their corresponding enable bits in the INTSTA register. Peripheral interrupts need either the global peripheral enable PEIE bit disabled, or the specific peripheral enable bit disabled. Disabling the peripherals via the global peripheral enable bit, disables all peripheral interrupts. GLINTD is set on RESET (interrupts disabled).

The `RETFIE` instruction clears the GLINTD bit while forcing the Program Counter (PC) to the value loaded at the Top-of-Stack.

When an interrupt is responded to, the GLINTD bit is automatically set to disable any further interrupt, the return address is pushed onto the stack and the PC is loaded with the interrupt vector. There are four interrupt vectors which help reduce interrupt latency.

The peripheral interrupt vector has multiple interrupt sources. Once in the peripheral Interrupt Service Routine, the source(s) of the interrupt can be determined by polling the interrupt flag bits. The peripheral interrupt flag bit(s) must be cleared in software before re-enabling interrupts to avoid continuous interrupts.

The PIC17C7XX devices have four interrupt vectors. These vectors and their hardware priority are shown in Table 6-1. If two enabled interrupts occur “at the same time”, the interrupt of the highest priority will be serviced first. This means that the vector address of that interrupt will be loaded into the program counter (PC).

**TABLE 6-1: INTERRUPT VECTORS/ PRIORITIES**

| Address | Vector | Priority |
|---|---|---|
| 0008h | External Interrupt on RA0/ INT pin (INTF) | 1 (Highest) |
| 0010h | TMR0 Overflow Interrupt (T0IF) | 2 |
| 0018h | External Interrupt on T0CKI (T0CKIF) | 3 |
| 0020h | Peripherals (PEIF) | 4 (Lowest) |

**Note 1:** Individual interrupt flag bits are set, regardless of the status of their corresponding mask bit or the GLINTD bit.

**2:** Before disabling any of the INTSTA enable bits, the GLINTD bit should be set (disabled).

## 6.5 RA0/INT Interrupt

The external interrupt on the RA0/INT pin is edge triggered. Either the rising edge if the INTEDG bit (T0STA<7>) is set, or the falling edge if the INTEDG bit is clear. When a valid edge appears on the RA0/INT pin, the INTF bit (INTSTA<4>) is set. This interrupt can be disabled by clearing the INTE control bit (INTSTA<0>). The INT interrupt can wake the processor from SLEEP. See Section 17.4 for details on SLEEP operation.

## 6.6 T0CKI Interrupt

The external interrupt on the RA1/T0CKI pin is edge triggered. Either the rising edge if the T0SE bit (T0STA<6>) is set, or the falling edge if the T0SE bit is clear. When a valid edge appears on the RA1/T0CKI pin, the T0CKIF bit (INTSTA<6>) is set. This interrupt can be disabled by clearing the T0CKIE control bit (INTSTA<2>). The T0CKI interrupt can wake up the processor from SLEEP. See Section 17.4 for details on SLEEP operation.

## 6.7 Peripheral Interrupt

The peripheral interrupt flag indicates that at least one of the peripheral interrupts occurred (PEIF is set). The PEIF bit is a read only bit and is a bit wise OR of all the flag bits in the PIR registers AND’d with the corresponding enable bits in the PIE registers. Some of the peripheral interrupts can wake the processor from SLEEP. See Section 17.4 for details on SLEEP operation.

## 6.8 Context Saving During Interrupts

During an interrupt, only the returned PC value is saved on the stack. Typically, users may wish to save key registers during an interrupt; e.g. WREG, ALUSTA and the BSR registers. This requires implementation in software.

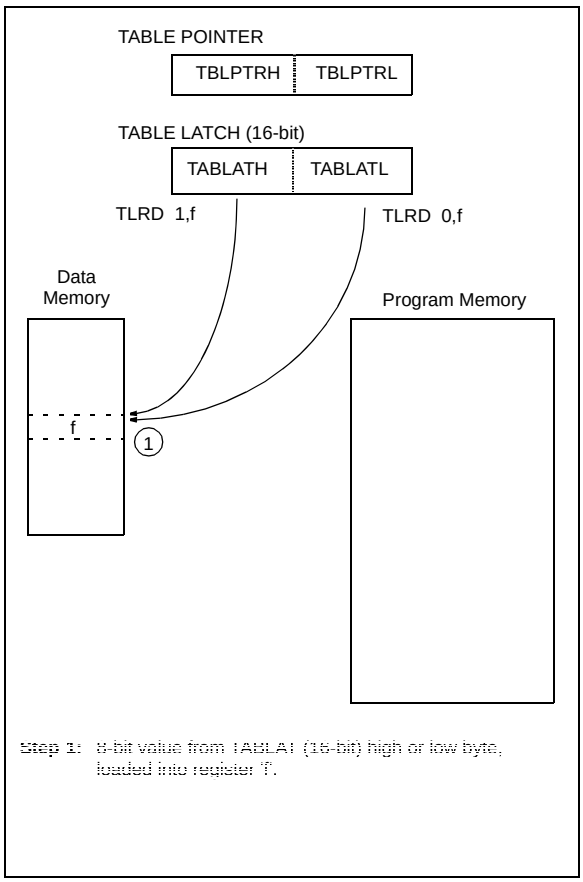
Example 6-2 shows the saving and restoring of information for an Interrupt Service Routine. This is for a simple interrupt scheme, where only one interrupt may occur at a time (no interrupt nesting). The SFRs are stored in the non-banked GPR area.

Example 6-2 shows the saving and restoring of information for a more complex Interrupt Service Routine. This is useful where nesting of interrupts is required. A maximum of 6 levels can be done by this example. The BSR is stored in the non-banked GPR area, while the other registers would be stored in a particular bank. Therefore, 6 saves may be done with this routine (since there are 6 non-banked GPR registers). These routines require a dedicated indirect addressing register, FSR0, to be selected for this.

The PUSH and POP code segments could either be in each Interrupt Service Routine, or could be subroutines that were called. Depending on the application, other registers may also need to be saved.

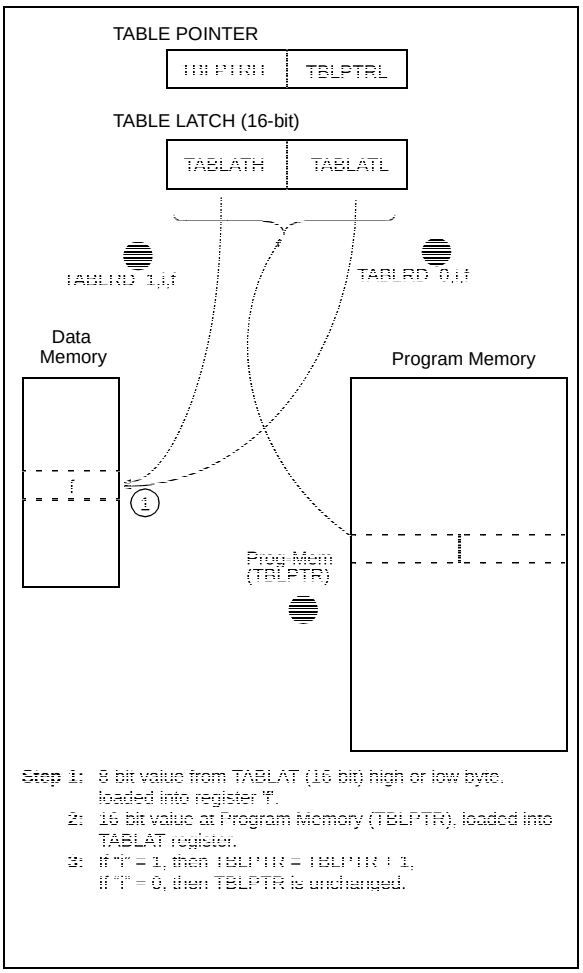
© 1998-2013 Microchip Technology Inc.

**FIGURE 8-3: `TLRD` INSTRUCTION OPERATION**




Step 1: 8-bit value from TABLAT (16-bit) high or low byte, loaded into register 'f'.

**FIGURE 8-4: `TABLRD` INSTRUCTION OPERATION**




Step 1: 8 bit value from TABLAT (16 bit) high or low byte, loaded into register 'f'.
2: 16 bit value at Program Memory (TBLPTR), loaded into TABLAT register.
3: If "t" = 1, then TBLPTR = TBLPTR + 1,
If "t" = 0, then TBLPTR is unchanged.


© 1998-2013 Microchip Technology Inc.

## 8.3 Table Reads

The table read allows the program memory to be read. This allows constants to be stored in the program memory space and retrieved into data memory when needed. Example 8-2 reads the 16-bit value at program memory address TBLPTR. After the dummy byte has been read from the TABLATH, the TABLATH is loaded with the 16-bit data from program memory address TBLPTR and then increments the TBLPTR value. The first read loads the data into the latch and can be considered a dummy read (unknown data loaded into 'f'). INDF0 should be configured for either auto-increment or auto-decrement.

**EXAMPLE 8-2: TABLE READ**

```
MOVLW   HIGH (TBL_ADDR) ; Load the Table
MOVWF   TBLPTRH         ;   address
MOVLW   LOW (TBL_ADDR)  ;
MOVWF   TBLPTRL         ;
TABLRD  0, 1, DUMMY   ; Dummy read,
                      ;  Updates TABLATH
                      ;  Increments TBLPTR
TLRD    1, INDF0      ; Read HI byte
                      ;   of TABLATH
TABLRD  0, 1, INDF0   ; Read LO byte
                      ;   of TABLATL and
                      ;   Update TABLATH
                      ;   Increment TBLPTR
```

**FIGURE 8-7: `TABLRD` TIMING**

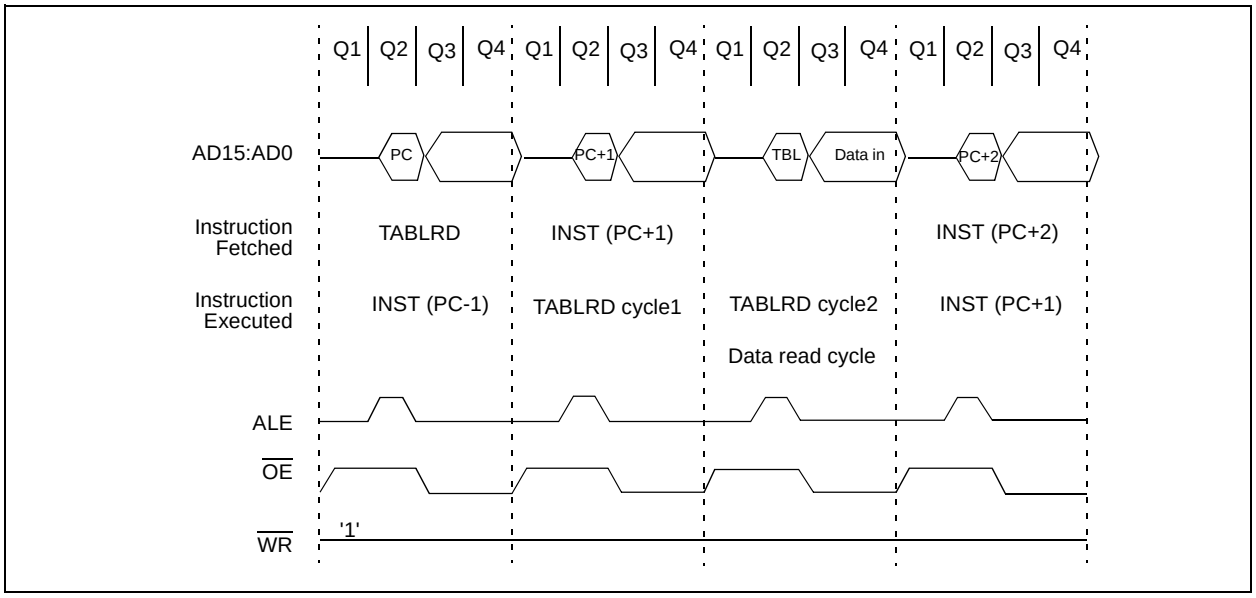
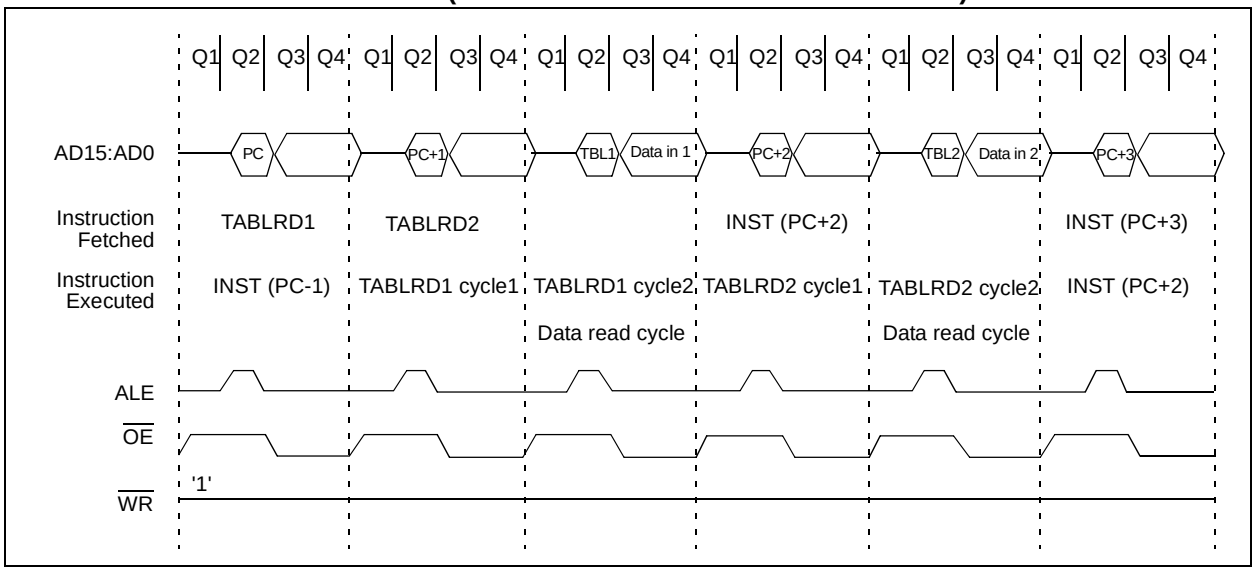



**FIGURE 8-8: `TABLRD` TIMING (CONSECUTIVE `TABLRD` INSTRUCTIONS)**





© 1998-2013 Microchip Technology Inc.

## 10.9 PORTJ and DDRJ Registers (PIC17C76X only)

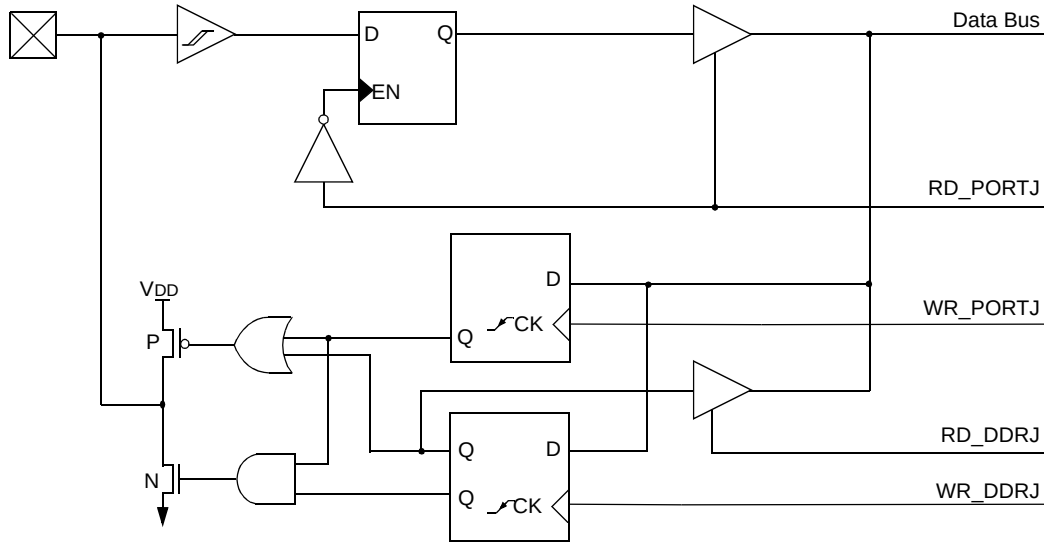
PORTJ is an 8-bit wide, bi-directional port. The corresponding data direction register is DDRJ. A '1' in DDRJ configures the corresponding port pin as an input. A '0' in the DDRJ register configures the corresponding port pin as an output. Reading PORTJ reads the status of the pins, whereas writing to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

PORTJ is a general purpose I/O port.

**EXAMPLE 10-9: INITIALIZING PORTJ**

```
MOVLB   8           ; Select Bank 8
CLRF    PORTJ, F    ; Initialize PORTJ data
                    ; latches before setting
                    ; the data direction
                    ; register
MOVLW    0xCF       ; Value used to initialize
                    ; data direction
MOVWF    DDRJ       ; Set RJ<3:0> as inputs
                    ; RJ<5:4> as outputs
                    ; RJ<7:6> as inputs
```

**FIGURE 10-19: PORTJ BLOCK DIAGRAM**




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

## 12.0 TIMER0

The Timer0 module consists of a 16-bit timer/counter, TMR0. The high byte is register TMR0H and the low byte is register TMR0L. A software programmable 8-bit prescaler makes Timer0 an effective 24-bit overflow timer. The clock source is software programmable as either the internal instruction clock, or an external clock on the RA1/T0CKI pin. The control bits for this module are in register T0STA (Figure 12-1).

**REGISTER 12-1: T0STA REGISTER (ADDRESS: 05h, UNBANKED)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | U-0 |
|---|---|---|---|---|---|---|---|
| INTEDG | T0SE | T0CS | T0PS3 | T0PS2 | T0PS1 | T0PS0 | — |
| bit 7 | | | | | | | bit 0 |

bit 7 **INTEDG**: RA0/INT Pin Interrupt Edge Select bit
This bit selects the edge upon which the interrupt is detected.
1 = Rising edge of RA0/INT pin generates interrupt
0 = Falling edge of RA0/INT pin generates interrupt

bit 6 **T0SE**: Timer0 Clock Input Edge Select bit
This bit selects the edge upon which TMR0 will increment.

When T0CS = 0 (External Clock):
1 = Rising edge of RA1/T0CKI pin increments TMR0 and/or sets the T0CKIF bit
0 = Falling edge of RA1/T0CKI pin increments TMR0 and/or sets the T0CKIF bit

When T0CS = 1 (Internal Clock):
Don't care

bit 5 **T0CS**: Timer0 Clock Source Select bit
This bit selects the clock source for TMR0.
1 = Internal instruction clock cycle ($T_{CY}$)
0 = External clock input on the T0CKI pin

bit 4-1 **T0PS3:T0PS0**: Timer0 Prescale Selection bits
These bits select the prescale value for TMR0.

| T0PS3:T0PS0 | Prescale Value |
|---|---|
| 0000 | 1:1 |
| 0001 | 1:2 |
| 0010 | 1:4 |
| 0011 | 1:8 |
| 0100 | 1:16 |
| 0101 | 1:32 |
| 0110 | 1:64 |
| 0111 | 1:128 |
| 1xxx | 1:256 |

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

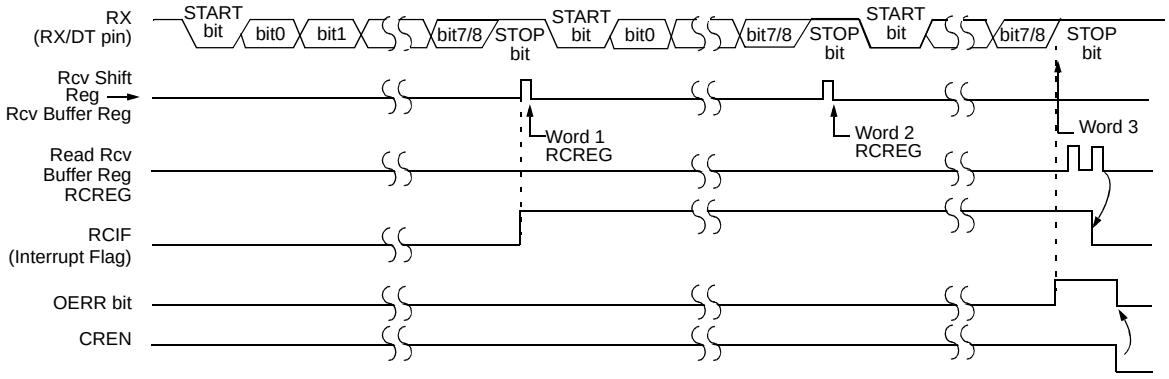
| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| - n = Value at POR Reset | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

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.


© 1998-2013 Microchip Technology Inc.

**TABLE 14-8: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER TRANSMISSION**

| 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 |
| 16h, Bank 0 | TXREG1 | TX7 | TX6 | TX5 | TX4 | TX3 | TX2 | TX1 | TX0 | 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 |
| 16h, Bank 4 | TXREG2 | TX7 | TX6 | TX5 | TX4 | TX3 | TX2 | TX1 | TX0 | 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 synchronous master transmission.

**FIGURE 14-8: SYNCHRONOUS TRANSMISSION**

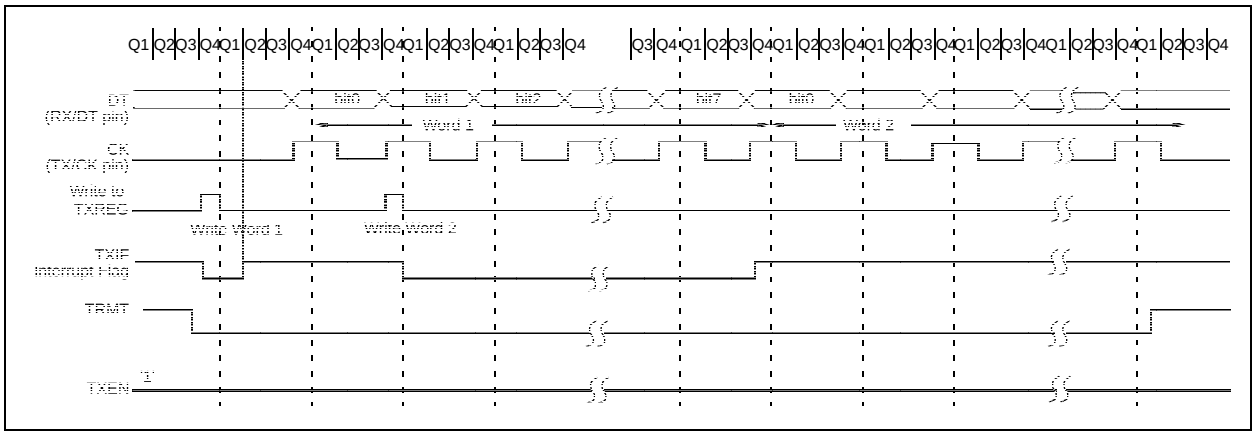
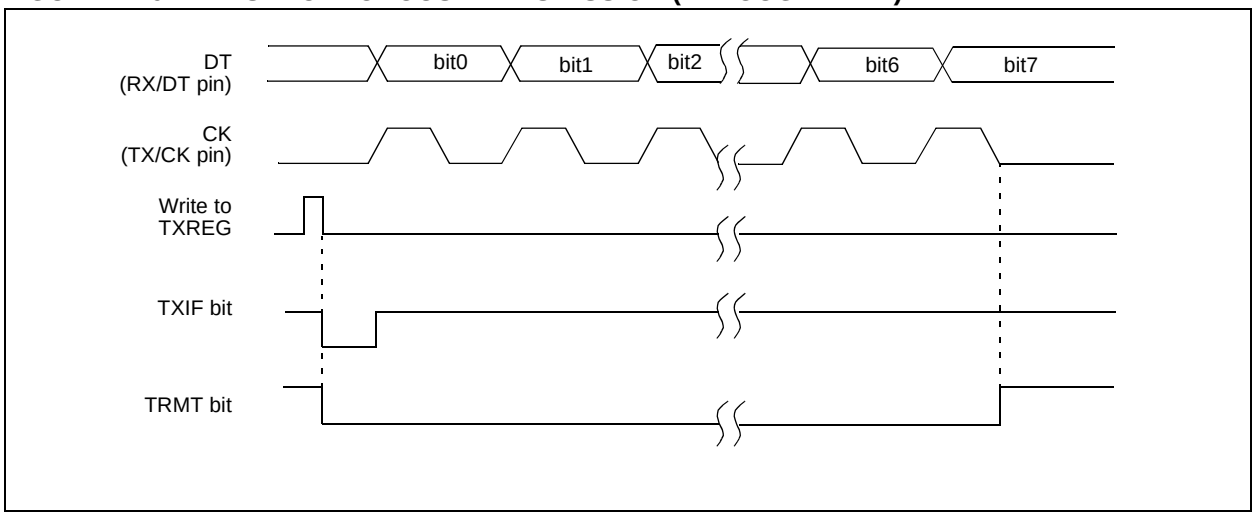



**FIGURE 14-9: SYNCHRONOUS TRANSMISSION (THROUGH TXEN)**





© 1998-2013 Microchip Technology Inc.

**REGISTER 15-2: SSPCON1: SYNC SERIAL PORT CONTROL REGISTER1 (ADDRESS 11h, BANK 6)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| WCOL | SSPOV | SSPEN | CKP | SSPM3 | SSPM2 | $\overline{\text{SSPM1}}$ | $\overline{\text{SSPM0}}$ |
| bit 7 | | | | | | | bit 0 |

bit 7 **WCOL**: Write Collision Detect bit

Master mode:
1 = A write to the SSPBUF register was attempted while the $I^2C$ conditions were not valid for a transmission to be started
0 = No collision

Slave mode:
1 = The SSPBUF 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

In SPI mode:
1 = A new byte is received while the SSPBUF register is still holding the previous data. In case of overflow, the data in SSPSR is lost. Overflow can only occur in Slave mode. In Slave mode, the user must read the SSPBUF, 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 SSPBUF register. (Must be cleared in software.)
0 = No overflow

In $I^2C$ mode:
1 = A byte is received while the SSPBUF 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, these pins must be properly configured as input or output.

In SPI mode:
1 = Enables serial port and configures SCK, SDO, SDI and $\overline{\text{SS}}$ as the source of the serial port pins
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
0 = Disables serial port and configures these pins as I/O port pins

**Note:** In SPI mode, these pins must be properly configured as input or output.

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

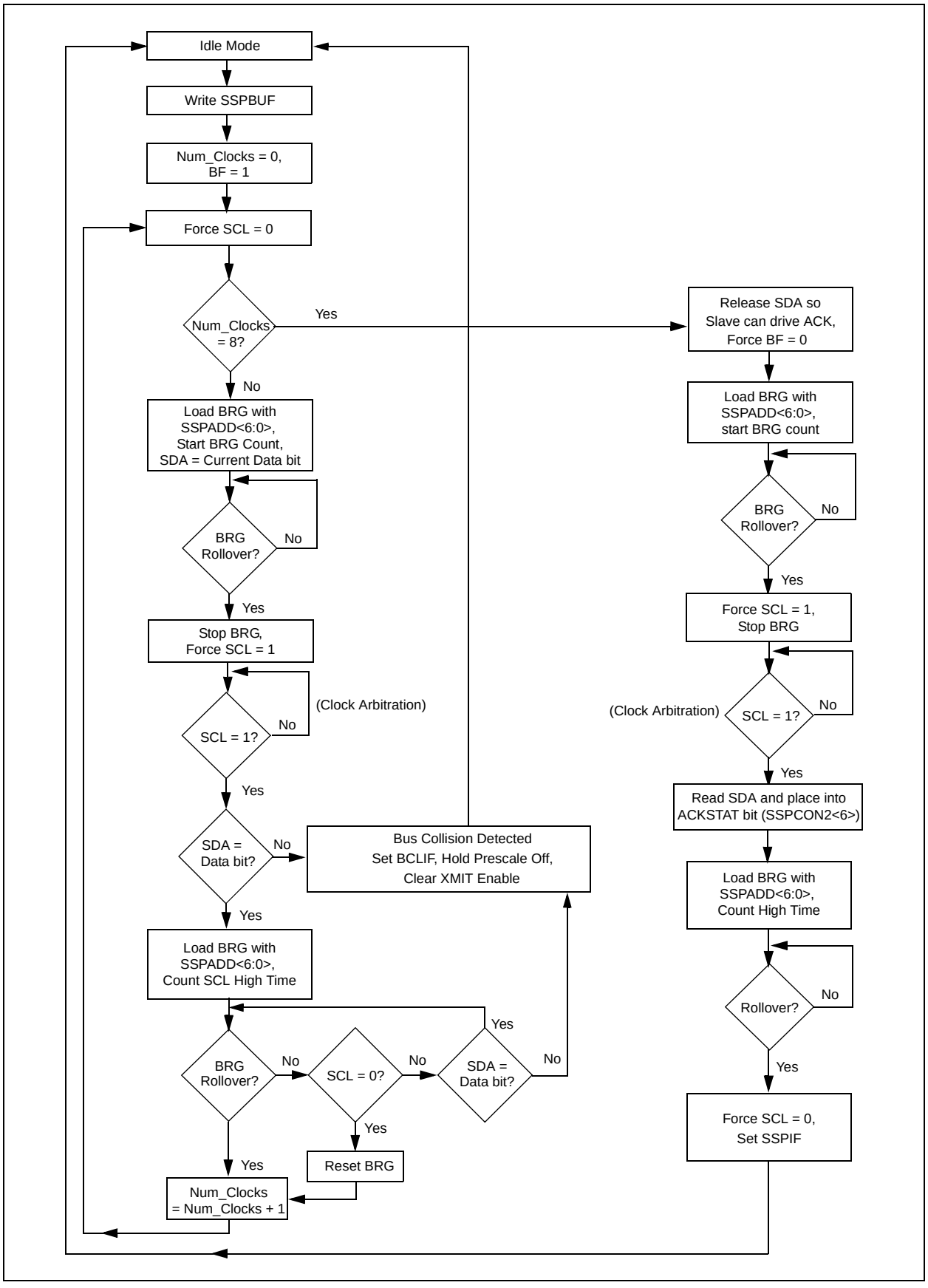
In $I^2C$ Slave mode:
SCK release control
1 = Enable clock
0 = Holds clock low (clock stretch). (Used to ensure data setup time.)

In $I^2C$ Master mode:
Unused in this mode

bit 3-0 **SSPM3:SSPM0**: Synchronous Serial Port Mode Select bits
0000 = SPI Master mode, clock = FOSC/4
0001 = SPI Master mode, clock = FOSC/16
0010 = SPI Master mode, clock = FOSC/64
0011 = SPI Master mode, clock = TMR2 output/2
0100 = SPI Slave mode, clock = SCK pin, $\overline{\text{SS}}$ pin control enabled
0101 = SPI Slave mode, clock = SCK pin, $\overline{\text{SS}}$ pin control disabled, $\overline{\text{SS}}$ can be used as I/O pin
0110 = $I^2C$ Slave mode, 7-bit address
0111 = $I^2C$ Slave mode, 10-bit address
1000 = $I^2C$ Master mode, clock = FOSC / (4 * (SSPADD+1) )
1xx1 = Reserved
1x1x = Reserved

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

**FIGURE 15-25: MASTER TRANSMIT FLOW CHART**





© 1998-2013 Microchip Technology Inc.

### 15.2.14 STOP CONDITION TIMING

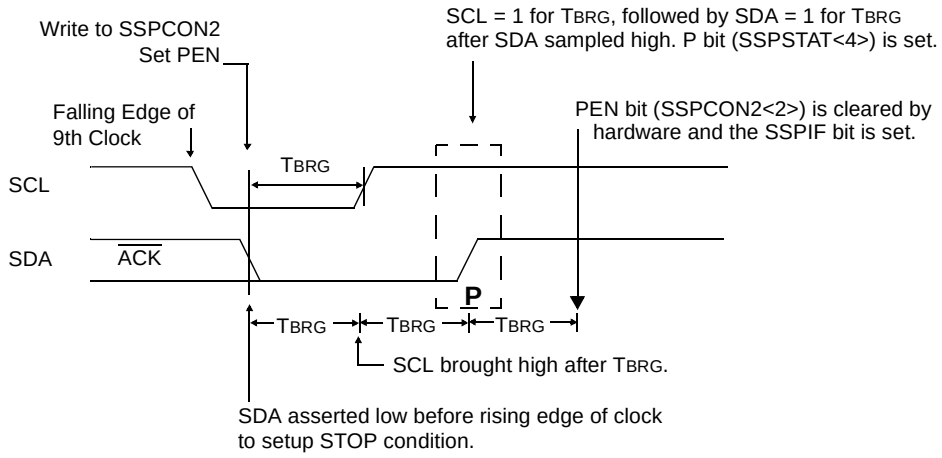
A STOP bit is asserted on the SDA pin at the end of a receive/transmit by setting the Stop Sequence Enable bit PEN (SSPCON2<2>). At the end of a receive/transmit the SCL line is held low after the falling edge of the ninth clock. When the PEN bit is set, the master will assert the SDA line low. When the SDA line is sampled low, the baud rate generator is reloaded and counts down to '0'. When the baud rate generator times out, the SCL pin will be brought high and one TBRG (baud rate generator rollover count) later, the SDA pin will be de-asserted. When the SDA pin is sampled high while SCL is high, the P bit (SSPSTAT<4>) is set. A TBRG later, the PEN bit is cleared and the SSPIF bit is set (Figure 15-31).

Whenever the firmware decides to take control of the bus, it will first determine if the bus is busy by checking the S and P bits in the SSPSTAT register. If the bus is busy, then the CPU can be interrupted (notified) when a STOP bit is detected (i.e., bus is free).

#### 15.2.14.1 WCOL Status Flag

If the user writes the SSPBUF when a STOP sequence is in progress, then WCOL is set and the contents of the buffer are unchanged (the write doesn't occur).

**FIGURE 15-31: STOP CONDITION RECEIVE OR TRANSMIT MODE**




**Note:** TBRG = one baud rate generator period.

**MOVPF** **Move p to f**

Syntax: [*label*] MOVPF p,f

Operands: $0 \le f \le 255$
$0 \le p \le 31$

Operation: (p) → (f)

Status Affected: Z

Encoding:

| | | | |
|---|---|---|---|
| 010p | pppp | ffff | ffff |

Description: Move data from data memory location 'p' to data memory location 'f'. Location 'f' can be anywhere in the 256 byte data space (00h to FFh), while 'p' can be 00h to 1Fh.

Either 'p' or 'f' can be WREG (a useful, special situation).

MOVPF is particularly useful for transferring a peripheral register (e.g. the timer or an I/O port) to a data memory location. Both 'f' and 'p' can be indirectly addressed.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example:
```
MOVPF   REG1, REG2
```
Before Instruction
REG1 = 0x11
REG2 = 0x33

After Instruction
REG1 = 0x11
REG2 = 0x11

**MOVWF** **Move WREG to f**

Syntax: [ *label* ] MOVWF f

Operands: $0 \le f \le 255$

Operation: (WREG) → (f)

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0001 | ffff | ffff |

Description: Move data from WREG to register 'f'. Location 'f' can be anywhere in the 256 byte data space.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example:
```
MOVWF   REG
```
Before Instruction
WREG = 0x4F
REG = 0xFF

After Instruction
WREG = 0x4F
REG = 0x4F


© 1998-2013 Microchip Technology Inc.

**SLEEP** **Enter SLEEP mode**

Syntax: [ *label* ] SLEEP

Operands: None

Operation: 00h → WDT;
0 → WDT postscaler;
1 → $\overline{TO}$;
0 → $\overline{PD}$

Status Affected: $\overline{TO}$, $\overline{PD}$

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0000 | 0011 |

Description: The power-down status bit ($\overline{PD}$) is cleared. The time-out status bit ($\overline{TO}$) is set. Watchdog Timer and its post-scaler are cleared.
The processor is put into SLEEP mode with the oscillator stopped.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | Process Data | Go to sleep |

Example: `SLEEP`

Before Instruction
$\overline{TO}$ = ?
$\overline{PD}$ = ?

After Instruction
$\overline{TO}$ = 1 †
$\overline{PD}$ = 0

† If WDT causes wake-up, this bit is cleared

**SUBLW** **Subtract WREG from Literal**

Syntax: [ *label* ] SUBLW k

Operands: $0 \le k \le 255$

Operation: k – (WREG) → (WREG)

Status Affected: OV, C, DC, Z

Encoding:

| | | | |
|---|---|---|---|
| 1011 | 0010 | kkkk | kkkk |

Description: WREG is subtracted from the eight-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 1: `SUBLW 0x02`

Before Instruction
WREG = 1
C = ?

After Instruction
WREG = 1
C = 1 ; result is positive
Z = 0

Example 2:

Before Instruction
WREG = 2
C = ?

After Instruction
WREG = 0
C = 1 ; result is zero
Z = 1

Example 3:

Before Instruction
WREG = 3
C = ?

After Instruction
WREG = FF ; (2's complement)
C = 0 ; result is negative
Z = 0

© 1998-2013 Microchip Technology Inc.

## 19.8 MPLAB ICD In-Circuit Debugger

Microchip's In-Circuit Debugger, MPLAB ICD, is a powerful, low cost, run-time development tool. This tool is based on the FLASH PIC16F87X and can be used to develop for this and other PIC microcontrollers from the PIC16CXXX family. The MPLAB ICD utilizes the in-circuit debugging capability built into the PIC16F87X. This feature, along with Microchip's In-Circuit Serial Programming™ protocol, offers cost-effective in-circuit FLASH debugging from the graphical user interface of the MPLAB Integrated Development Environment. This enables a designer to develop and debug source code by watching variables, single-stepping and setting break points. Running at full speed enables testing hardware in real-time.

## 19.9 PRO MATE II Universal Device Programmer

The PRO MATE II universal device programmer is a full-featured programmer, capable of operating in stand-alone mode, as well as PC-hosted mode. The PRO MATE II device programmer is CE compliant.

The PRO MATE II device programmer has programmable $V_{DD}$ and $V_{PP}$ supplies, which allow it to verify programmed memory at $V_{DD}$ min and $V_{DD}$ max for maximum reliability. It has an LCD display for instructions and error messages, keys to enter commands and a modular detachable socket assembly to support various package types. In stand-alone mode, the PRO MATE II device programmer can read, verify, or program PIC MCU devices. It can also set code protection in this mode.

## 19.10 PICSTART Plus Entry Level Development Programmer

The PICSTART Plus development programmer is an easy-to-use, low cost, prototype programmer. It connects to the PC via a COM (RS-232) port. MPLAB Integrated Development Environment software makes using the programmer simple and efficient.

The PICSTART Plus development programmer supports all PIC devices with up to 40 pins. Larger pin count devices, such as the PIC16C92X and PIC17C76X, may be supported with an adapter socket. The PICSTART Plus development programmer is CE compliant.

## 19.11 PICDEM 1 Low Cost PIC MCU Demonstration Board

The PICDEM 1 demonstration board is a simple board which demonstrates the capabilities of several of Microchip's microcontrollers. The microcontrollers supported are: PIC16C5X (PIC16C54 to PIC16C58A), PIC16C61, PIC16C62X, PIC16C71, PIC16C8X, PIC17C42, PIC17C43 and PIC17C44. All necessary hardware and software is included to run basic demo programs. The user can program the sample microcontrollers provided with the PICDEM 1 demonstration board on a PRO MATE II device programmer, or a PICSTART Plus development programmer, and easily test firmware. The user can also connect the PICDEM 1 demonstration board to the MPLAB ICE in-circuit emulator and download the firmware to the emulator for testing. A prototype area is available for the user to build some additional hardware and connect it to the microcontroller socket(s). Some of the features include an RS-232 interface, a potentiometer for simulated analog input, push button switches and eight LEDs connected to PORTB.

## 19.12 PICDEM 2 Low Cost PIC16CXX Demonstration Board

The PICDEM 2 demonstration board is a simple demonstration board that supports the PIC16C62, PIC16C64, PIC16C65, PIC16C73 and PIC16C74 microcontrollers. All the necessary hardware and software is included to run the basic demonstration programs. The user can program the sample microcontrollers provided with the PICDEM 2 demonstration board on a PRO MATE II device programmer, or a PICSTART Plus development programmer, and easily test firmware. The MPLAB ICE in-circuit emulator may also be used with the PICDEM 2 demonstration board to test firmware. A prototype area has been provided to the user for adding additional hardware and connecting it to the microcontroller socket(s). Some of the features include a RS-232 interface, push button switches, a potentiometer for simulated analog input, a serial EEPROM to demonstrate usage of the $I^2C$™ bus and separate headers for connection to an LCD module and a keypad.

**TABLE 19-1: DEVELOPMENT TOOLS FROM MICROCHIP**

| | | PIC12CXXX | PIC14000 | PIC16C5X | PIC16C6X | PIC16CXXX | PIC16F62X | PIC16C7X | PIC16C7XX | PIC16C8X | PIC16F8XX | PIC16C9XX | PIC17C4X | PIC17C7XX | PIC18CXX2 | 24CXX/ 25CXX/ 93CXX | HCSXXX | MCRFXXX | MCP2510 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Software Tools | MPLAB® Integrated Development Environment | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | | |
| | MPLAB® C17 C Compiler | | | | | | | | | | | | ✓ | ✓ | | | | | |
| | MPLAB® C18 C Compiler | | | | | | | | | | | | | | ✓ | | | | |
| | MPASM™ Assembler/ MPLINK™ Object Linker | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | |
| Emulators | MPLAB® ICE In-Circuit Emulator | ✓ | ✓ | ✓ | ✓ | ✓ | ✓** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | | |
| | ICEPIC™ In-Circuit Emulator | ✓ | | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | | ✓ | | | | | | | |
| Debugger | MPLAB® ICD In-Circuit Debugger | | | | ✓* | | | ✓* | | | ✓ | | | | | | | | |
| Programmers | PICSTART® Plus Entry Level Development Programmer | ✓ | ✓ | ✓ | ✓ | ✓ | ✓** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | | |
| | PRO MATE® II Universal Device Programmer | ✓ | ✓ | ✓ | ✓ | ✓ | ✓** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | |
| Demo Boards and Eval Kits | PICDEM™ 1 Demonstration Board | | | ✓ | | ✓ | | ✓† | | ✓ | | | ✓ | | | | | | |
| | PICDEM™ 2 Demonstration Board | | | | ✓† | | | ✓† | | | | | | | ✓ | | | | |
| | PICDEM™ 3 Demonstration Board | | | | | | | | | | | ✓ | | | | | | | |
| | PICDEM™ 14A Demonstration Board | | ✓ | | | | | | | | | | | | | | | | |
| | PICDEM™ 17 Demonstration Board | | | | | | | | | | | | | ✓ | | | | | |
| | KEELOQ® Evaluation Kit | | | | | | | | | | | | | | | | ✓ | | |
| | KEELOQ® Transponder Kit | | | | | | | | | | | | | | | | ✓ | | |
| | microID™ Programmer's Kit | | | | | | | | | | | | | | | | | ✓ | |
| | 125 kHz microID™ Developer's Kit | | | | | | | | | | | | | | | | | ✓ | |
| | 125 kHz Anticollision microID™ Developer's Kit | | | | | | | | | | | | | | | | | ✓ | |
| | 13.56 MHz Anticollision microID™ Developer's Kit | | | | | | | | | | | | | | | | | ✓ | |
| | MCP2510 CAN Developer's Kit | | | | | | | | | | | | | | | | | | ✓ |

* Contact the Microchip Technology Inc. web site at www.microchip.com for information on how to use the MPLAB® ICD In-Circuit Debugger (DV164001) with PIC16C62, 63, 64, 65, 72, 73, 74, 76, 77.
** Contact Microchip Technology Inc. for availability date.
† Development tool is available on select devices.

## READER RESPONSE

It is our intention to provide you with the best documentation possible to ensure successful use of your Microchip product. If you wish to provide your comments on organization, clarity, subject matter, and ways in which our documentation can better serve you, please FAX your comments to the Technical Publications Manager at (480) 792-4150.

Please list the following information, and use this outline to provide us with your comments about this Data Sheet.

To: Technical Publications Manager Total Pages Sent

RE: Reader Response

From: Name _____

Company _____

Address _____

City / State / ZIP / Country _____

Telephone: (_____) _____ - _____ FAX: (_____) _____ - _____

Application (optional):

Would you like a reply? ___Y ___N

Device: **PIC17C7XX** Literature Number: **DS30289C**

Questions:

1. What are the best features of this document?

2. How does this document meet your hardware and software development needs?

3. Do you find the organization of this data sheet easy to follow? If not, why?

4. What additions to the data sheet do you think would enhance the structure and subject?

5. What deletions from the data sheet could be made without affecting the overall usefulness?

6. Is there any incorrect or misleading information (what and where)?

7. How would you improve this document?

8. How would you improve our software, systems, and silicon products?


© 1998-2013 Microchip Technology Inc.

## PRODUCT IDENTIFICATION SYSTEM

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.

| PART NO. | X | /XX | XXX |
|---|---|---|---|
| Device | Temperature Range | Package | Pattern |

| | |
|---|---|
| Device | PIC17C756: Standard VDD range<br>PIC17C756T: (Tape and Reel)<br>PIC17LC756: Extended VDD range |
| Temperature Range | - = 0°C to +70°C<br>I = -40°C to +85°C |
| Package | CL = Windowed LCC<br>PT = TQFP<br>L = PLCC |
| Pattern | QTP, SQTP, ROM Code (factory specified) or Special Requirements . Blamk for OTP and Windowed devices. |

**Examples:**

a) PIC17C756 – 16L Commercial Temp., PLCC package, 16 MHz, normal VDD limits

b) PIC17LC756–08/PT Commercial Temp., TQFP package, 8MHz, extended VDD limits

c) PIC17C756–33I/PT Industrial Temp., TQFP package, 33 MHz, normal VDD limits

* JW Devices are UV erasable and can be programmed to any device configuration. JW Devices meet the electrical requirement of each oscillator type.

### Sales and Support

**Data Sheets**
Products supported by a preliminary Data Sheet may have an errata sheet describing minor operational differences and recommended workarounds. To determine if an errata sheet exists for a particular device, please contact one of the following:

1. Your local Microchip sales office
2. The Microchip Corporate Literature Center U.S. FAX: (480) 792-7277
3. The Microchip Worldwide Site (www.microchip.com)

Please specify which device, revision of silicon and Data Sheet (include Literature #) you are using.

**New Customer Notification System**
Register on our web site (www.microchip.com/cn) to receive the most current information on our products.

© 1998-2013 Microchip Technology Inc.