Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

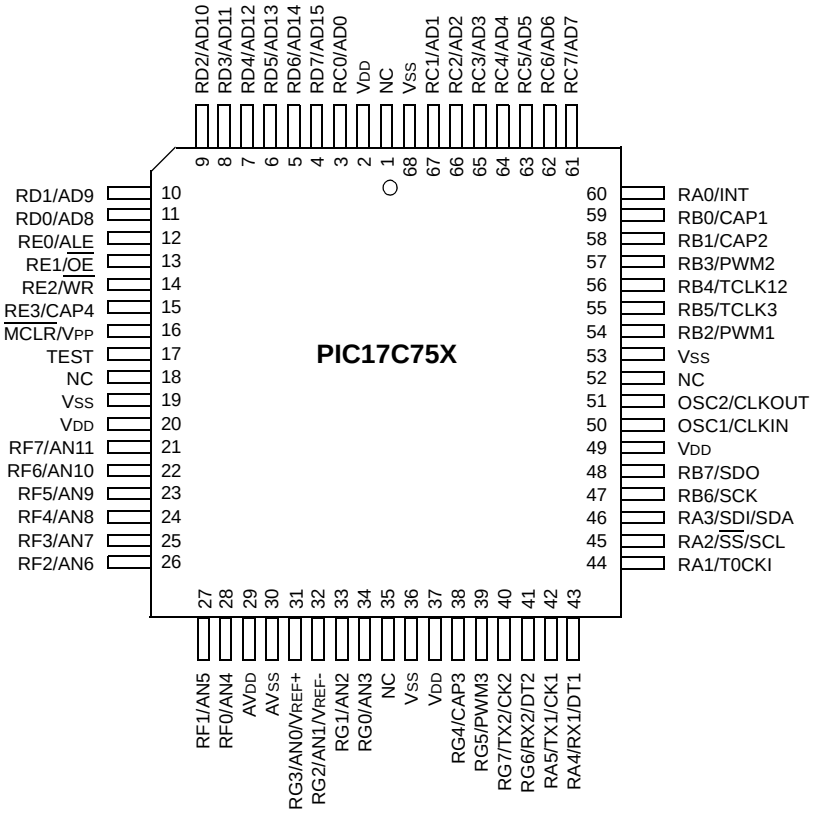
"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded"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

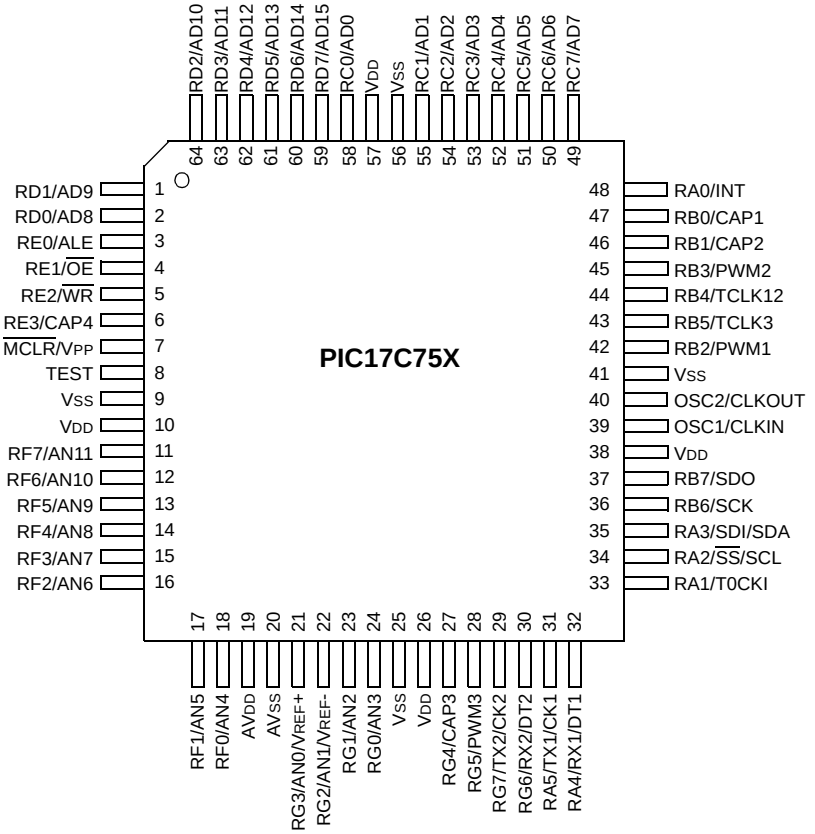
Details	
Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	33MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	50
Program Memory Size	32KB (16K x 16)
Program Memory Type	OTP
EEPROM Size	-
RAM Size	902 x 8
Voltage - Supply (Vcc/Vdd)	4.5V ~ 5.5V
Data Converters	A/D 12x10b
Oscillator Type	External
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	68-LCC (J-Lead)
Supplier Device Package	68-PLCC (24.23x24.23)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic17c756a-33e-l

Pin Diagrams cont.'d

68-Pin PLCC



64-Pin TQFP



4.0 ON-CHIP OSCILLATOR CIRCUIT

The internal oscillator circuit is used to generate the device clock. Four device clock periods generate an internal instruction clock (T_{CY}).

There are four modes that the oscillator can operate in. They are selected by the device configuration bits during device programming. These modes are:

- LF Low Frequency ($F_{OSC} \leq 2$ MHz)
- XT Standard Crystal/Resonator Frequency (2 MHz $\leq F_{OSC} \leq$ 33 MHz)
- EC External Clock Input (Default oscillator configuration)
- RC External Resistor/Capacitor ($F_{OSC} \leq 4$ MHz)

There are two timers that offer necessary delays on power-up. One is the Oscillator Start-up Timer (OST), intended to keep the chip in RESET until the crystal oscillator is stable. The other is the Power-up Timer (PWRT), which provides a fixed delay of 96 ms (nominal) on POR and BOR. The PWRT is designed to keep the part in RESET while the power supply stabilizes. With these two timers on-chip, most applications need no external RESET circuitry.

SLEEP mode is designed to offer a very low current power-down mode. The user can wake from SLEEP through external RESET, Watchdog Timer Reset, or through an interrupt.

Several oscillator options are made available to allow the part to better fit the application. The RC oscillator option saves system cost while the LF crystal option saves power. Configuration bits are used to select various options.

4.1 Oscillator Configurations

4.1.1 OSCILLATOR TYPES

The PIC17CXXX can be operated in four different oscillator modes. The user can program two configuration bits (FOSC1:FOSC0) to select one of these four modes:

- LF Low Power Crystal
- XT Crystal/Resonator
- EC External Clock Input
- RC Resistor/Capacitor

The main difference between the LF and XT modes is the gain of the internal inverter of the oscillator circuit, which allows the different frequency ranges.

For more details on the device configuration bits, see Section 17.0.

4.1.2 CRYSTAL OSCILLATOR/CERAMIC RESONATORS

In XT or LF modes, a crystal or ceramic resonator is connected to the OSC1/CLKIN and OSC2/CLKOUT pins to establish oscillation (Figure 4-2). The PIC17CXXX oscillator design requires the use of a parallel cut crystal. Use of a series cut crystal may give a frequency out of the crystal manufacturers specifications.

For frequencies above 24 MHz, it is common for the crystal to be an overtone mode crystal. Use of overtone mode crystals require a tank circuit to attenuate the gain at the fundamental frequency. Figure 4-3 shows an example circuit.

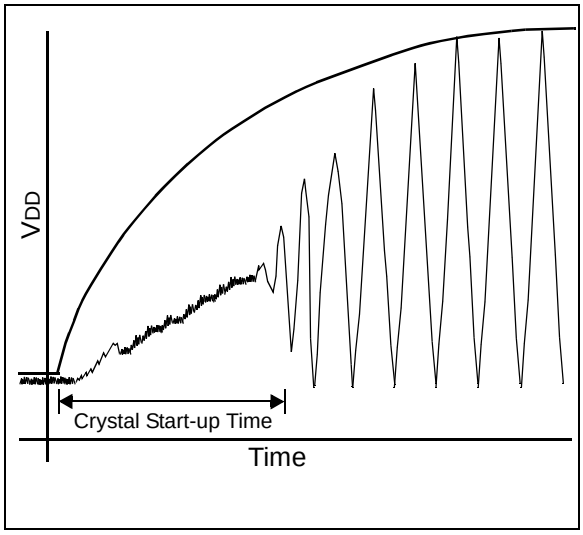
4.1.3 OSCILLATOR/RESONATOR START-UP

As the device voltage increases from Vss, the oscillator will start its oscillations. The time required for the oscillator to start oscillating depends on many factors. These include:

- Crystal/resonator frequency
- Capacitor values used (C1 and C2)
- Device V_{DD} rise time
- System temperature
- Series resistor value (and type) if used
- Oscillator mode selection of device (which selects the gain of the internal oscillator inverter)

Figure 4-1 shows an example of a typical oscillator/resonator start-up. The peak-to-peak voltage of the oscillator waveform can be quite low (less than 50% of device V_{DD}) when the waveform is centered at $V_{DD}/2$ (refer to parameter #D033 and parameter #D043 in the electrical specification section).

FIGURE 4-1: OSCILLATOR/ RESONATOR START-UP CHARACTERISTICS



NOTES:

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.

8.1 Table Writes to Internal Memory

A table write operation to internal memory causes a long write operation. The long write is necessary for programming the internal EPROM. Instruction execution is halted while in a long write cycle. The long write will be terminated by any enabled interrupt. To ensure that the EPROM location has been well programmed, a minimum programming time is required (see specification #D114). Having only one interrupt enabled to terminate the long write ensures that no unintentional interrupts will prematurely terminate the long write.

The sequence of events for programming an internal program memory location should be:

1. Disable all interrupt sources, except the source to terminate EPROM program write.
2. Raise $\overline{MCLR}/V_{PP}$ pin to the programming voltage.
3. Clear the WDT.
4. Do the table write. The interrupt will terminate the long write.
5. Verify the memory location (table read).

Note 1: Programming requirements must be met. See timing specification in electrical specifications for the desired device. Violating these specifications (including temperature) may result in EPROM locations that are not fully programmed and may lose their state over time.

2: If the V_{PP} requirement is not met, the table write is a 2-cycle write and the program memory is unchanged.

8.1.1 TERMINATING LONG WRITES

An interrupt source or RESET are the only events that terminate a long write operation. Terminating the long write from an interrupt source requires that the interrupt enable and flag bits are set. The GLINTD bit only enables the vectoring to the interrupt address.

If the T0CKI, RA0/INT, or TMR0 interrupt source is used to terminate the long write, the interrupt flag of the highest priority enabled interrupt, will terminate the long write and automatically be cleared.

Note 1: If an interrupt is pending, the `TABLWT` is aborted (a `NOP` is executed). The highest priority pending interrupt, from the T0CKI, RA0/INT, or TMR0 sources that is enabled, has its flag cleared.

2: If the interrupt is not being used for the program write timing, the interrupt should be disabled. This will ensure that the interrupt is not lost, nor will it terminate the long write prematurely.

If a peripheral interrupt source is used to terminate the long write, the interrupt enable and flag bits must be set. The interrupt flag will not be automatically cleared upon the vectoring to the interrupt vector address.

The GLINTD bit determines whether the program will branch to the interrupt vector when the long write is terminated. If GLINTD is clear, the program will vector, if GLINTD is set, the program will not vector to the interrupt address.

TABLE 8-1: INTERRUPT - TABLE WRITE INTERACTION

Interrupt Source	GLINTD	Enable Bit	Flag Bit	Action
RA0/INT, TMR0, T0CKI	0	1	1	Terminate long table write (to internal program memory), branch to interrupt vector (branch clears flag bit).
	0	1	0	None.
	1	0	x	None.
	1	1	1	Terminate long table write, do not branch to interrupt vector (flag is automatically cleared).
Peripheral	0	1	1	Terminate long table write, branch to interrupt vector.
	0	1	0	None.
	1	0	x	None.
	1	1	1	Terminate long table write, do not branch to interrupt vector (flag remains set).

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.

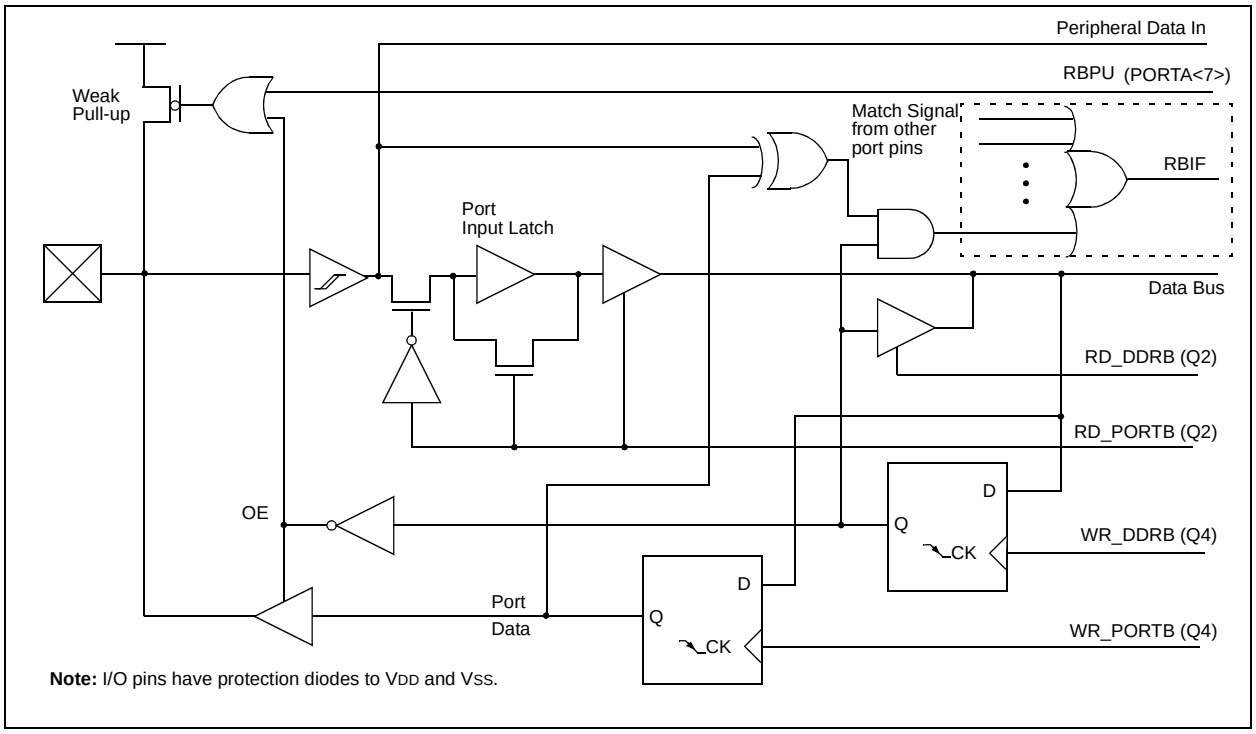
A mismatch condition will continue to set the RBIF bit. Reading, then writing PORTB, will end the mismatch condition and allow the RBIF bit to be cleared.

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



12.1 Timer0 Operation

When the T0CS (T0STA<5>) bit is set, TMR0 increments on the internal clock. When T0CS is clear, TMR0 increments on the external clock (RA1/T0CKI pin). The external clock edge can be selected in software. When the T0SE (T0STA<6>) bit is set, the timer will increment on the rising edge of the RA1/T0CKI pin. When T0SE is clear, the timer will increment on the falling edge of the RA1/T0CKI pin. The prescaler can be programmed to introduce a prescale of 1:1 to 1:256. The timer increments from 0000h to FFFFh and rolls over to 0000h. On overflow, the TMR0 Interrupt Flag bit (T0IF) is set. The TMR0 interrupt can be masked by clearing the corresponding TMR0 Interrupt Enable bit (T0IE). The TMR0 Interrupt Flag bit (T0IF) is automatically cleared when vectoring to the TMR0 interrupt vector.

12.2 Using Timer0 with External Clock

When an external clock input is used for Timer0, it is synchronized with the internal phase clocks. Figure 12-2 shows the synchronization of the external clock. This synchronization is done after the prescaler. The output of the prescaler (PSOUT) is sampled twice in every instruction cycle to detect a rising or a falling edge. The timing requirements for the external clock are detailed in the electrical specification section.

12.2.1 DELAY FROM EXTERNAL CLOCK EDGE

Since the prescaler output is synchronized with the internal clocks, there is a small delay from the time the external clock edge occurs to the time TMR0 is actually incremented. Figure 12-2 shows that this delay is between 3TOSC and 7TOSC. Thus, for example, measuring the interval between two edges (e.g. period) will be accurate within ±4TOSC (±121 ns @ 33 MHz).

FIGURE 12-1: TIMER0 MODULE BLOCK DIAGRAM

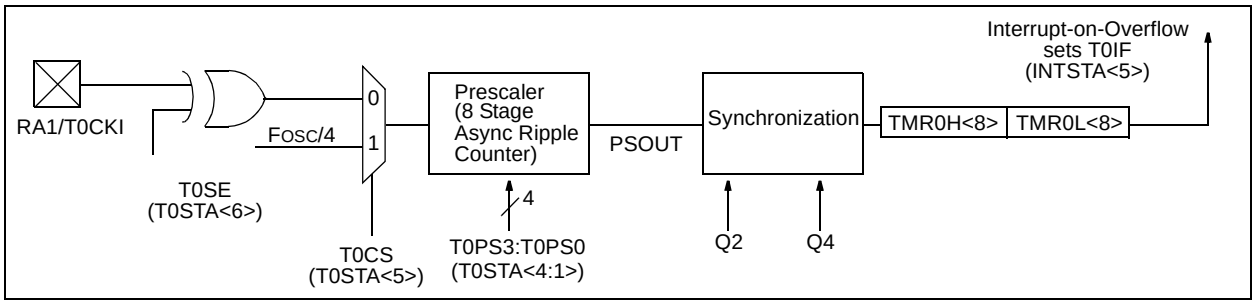
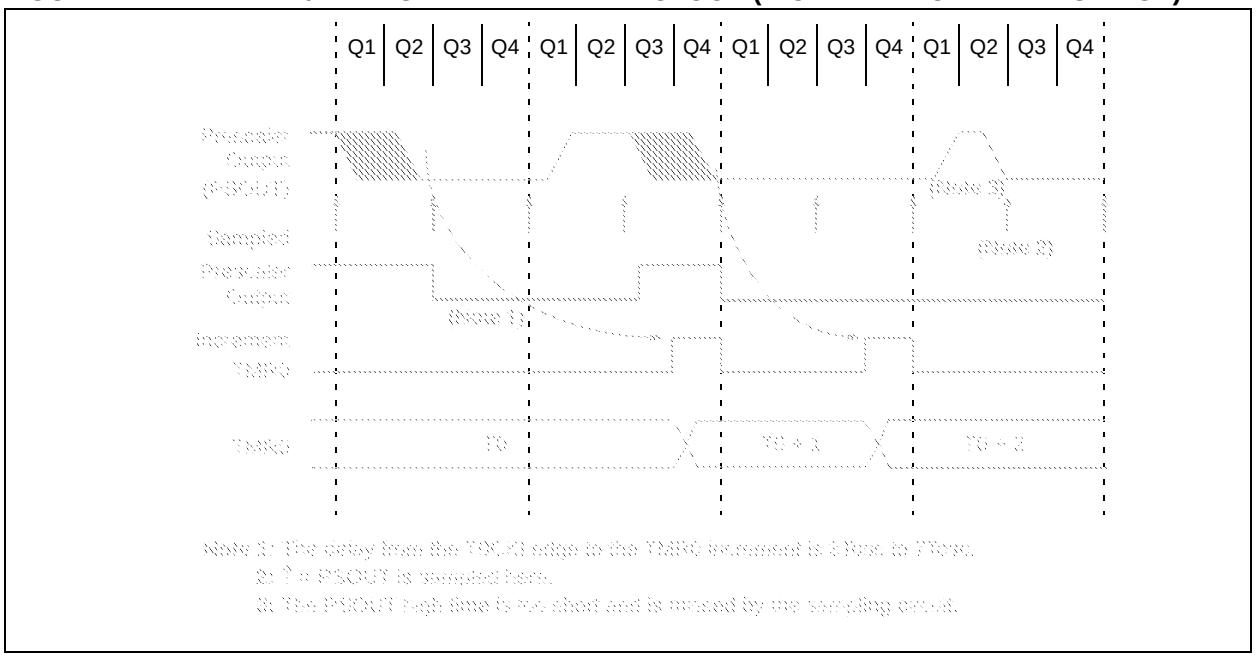


FIGURE 12-2: TMR0 TIMING WITH EXTERNAL CLOCK (INCREMENT ON FALLING EDGE)



13.2.2 FOUR CAPTURE MODE

This mode is selected by setting bit CA1/$\overline{PR3}$. A block diagram is shown in Figure 13-6. In this mode, TMR3 runs without a period register and increments from 0000h to FFFFh and rolls over to 0000h. The TMR3 interrupt Flag (TMR3IF) is set on this rollover. The TMR3IF bit must be cleared in software.

Registers PR3H/CA1H and PR3L/CA1L make a 16-bit capture register (Capture1). It captures events on pin RB0/CAP1. Capture mode is configured by the CA1ED1 and CA1ED0 bits. Capture1 Interrupt Flag bit (CA1IF) is set upon detection of the capture event. The corresponding interrupt mask bit is CA1IE. The Capture1 Overflow Status bit is CA1OVF.

All the captures operate in the same manner. Refer to Section 13.2.1 for the operation of capture.

FIGURE 13-6: TIMER3 WITH FOUR CAPTURES BLOCK DIAGRAM

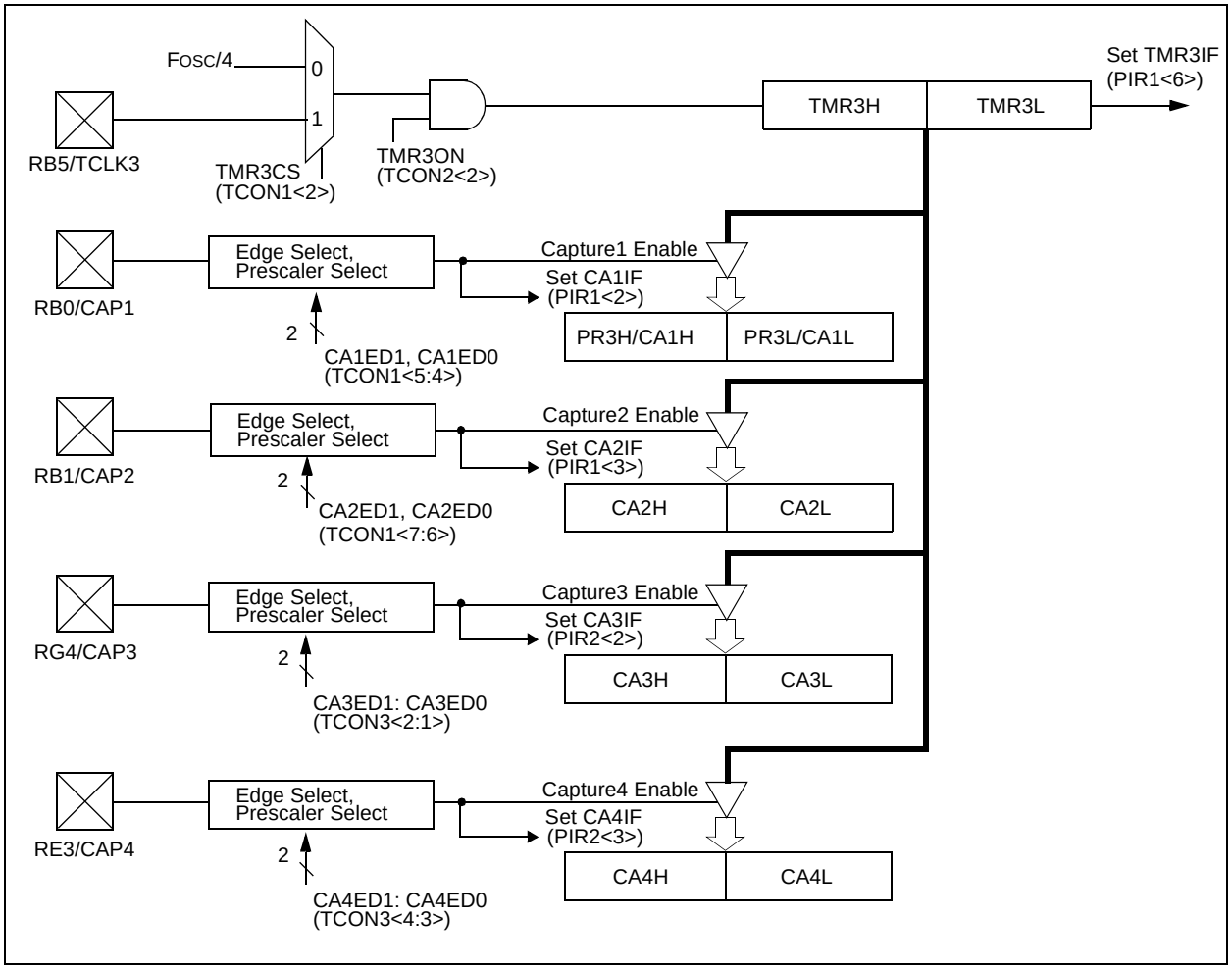


TABLE 14-4: BAUD RATES FOR SYNCHRONOUS MODE

BAUD RATE (K)	FOSC = 33 MHz			FOSC = 25 MHz			FOSC = 20 MHz			FOSC = 16 MHz		
	KBAUD	%ERROR	SPBRG VALUE (DECIMAL)	KBAUD	%ERROR	SPBRG VALUE (DECIMAL)	KBAUD	%ERROR	SPBRG VALUE (DECIMAL)	KBAUD	%ERROR	SPBRG VALUE (DECIMAL)
0.3	NA	—	—	NA	—	—	NA	—	—	NA	—	—
1.2	NA	—	—	NA	—	—	NA	—	—	NA	—	—
2.4	NA	—	—	NA	—	—	NA	—	—	NA	—	—
9.6	NA	—	—	NA	—	—	NA	—	—	NA	—	—
19.2	NA	—	—	NA	—	—	19.53	+1.73	255	19.23	+0.16	207
76.8	77.10	+0.39	106	77.16	+0.47	80	76.92	+0.16	64	76.92	+0.16	51
96	95.93	-0.07	85	96.15	+0.16	64	96.15	+0.16	51	95.24	-0.79	41
300	294.64	-1.79	27	297.62	-0.79	20	294.1	-1.96	16	307.69	+2.56	12
500	485.29	-2.94	16	480.77	-3.85	12	500	0	9	500	0	7
HIGH	8250	—	0	6250	—	0	5000	—	0	4000	—	0
LOW	32.22	—	255	24.41	—	255	19.53	—	255	15.625	—	255

BAUD RATE (K)	FOSC = 10 MHz			FOSC = 7.159 MHz			FOSC = 5.068 MHz		
	KBAUD	%ERROR	SPBRG VALUE (DECIMAL)	KBAUD	%ERROR	SPBRG VALUE (DECIMAL)	KBAUD	%ERROR	SPBRG VALUE (DECIMAL)
0.3	NA	—	—	NA	—	—	NA	—	—
1.2	NA	—	—	NA	—	—	NA	—	—
2.4	NA	—	—	NA	—	—	NA	—	—
9.6	9.766	+1.73	255	9.622	+0.23	185	9.6	0	131
19.2	19.23	+0.16	129	19.24	+0.23	92	19.2	0	65
76.8	75.76	-1.36	32	77.82	+1.32	22	79.2	+3.13	15
96	96.15	+0.16	25	94.20	-1.88	18	97.48	+1.54	12
300	312.5	+4.17	7	298.3	-0.57	5	316.8	+5.60	3
500	500	0	4	NA	—	—	NA	—	—
HIGH	2500	—	0	1789.8	—	0	1267	—	0
LOW	9.766	—	255	6.991	—	255	4.950	—	255

BAUD RATE (K)	FOSC = 3.579 MHz			FOSC = 1 MHz			FOSC = 32.768 kHz		
	KBAUD	%ERROR	SPBRG VALUE (DECIMAL)	KBAUD	%ERROR	SPBRG VALUE (DECIMAL)	KBAUD	%ERROR	SPBRG VALUE (DECIMAL)
0.3	NA	—	—	NA	—	—	0.303	+1.14	26
1.2	NA	—	—	1.202	+0.16	207	1.170	-2.48	6
2.4	NA	—	—	2.404	+0.16	103	NA	—	—
9.6	9.622	+0.23	92	9.615	+0.16	25	NA	—	—
19.2	19.04	-0.83	46	19.24	+0.16	12	NA	—	—
76.8	74.57	-2.90	11	83.34	+8.51	2	NA	—	—
96	99.43	_3.57	8	NA	—	—	NA	—	—
300	298.3	-0.57	2	NA	—	—	NA	—	—
500	NA	—	—	NA	—	—	NA	—	—
HIGH	894.9	—	0	250	—	0	8.192	—	0
LOW	3.496	—	255	0.976	—	255	0.032	—	255

15.0 MASTER SYNCHRONOUS SERIAL PORT (MSSP) MODULE

The Master Synchronous Serial Port (MSSP) module is a serial interface useful for communicating with other peripheral or microcontroller devices. These peripheral devices may be serial EEPROMs, shift registers, display drivers, A/D converters, etc. The MSSP module can operate in one of two modes:

- Serial Peripheral Interface (SPI)
- Inter-Integrated Circuit™ (I^2C)

Figure 15-1 shows a block diagram for the SPI mode, while Figure 15-2 and Figure 15-3 show the block diagrams for the two different I^2C modes of operation.

FIGURE 15-1: SPI MODE BLOCK DIAGRAM

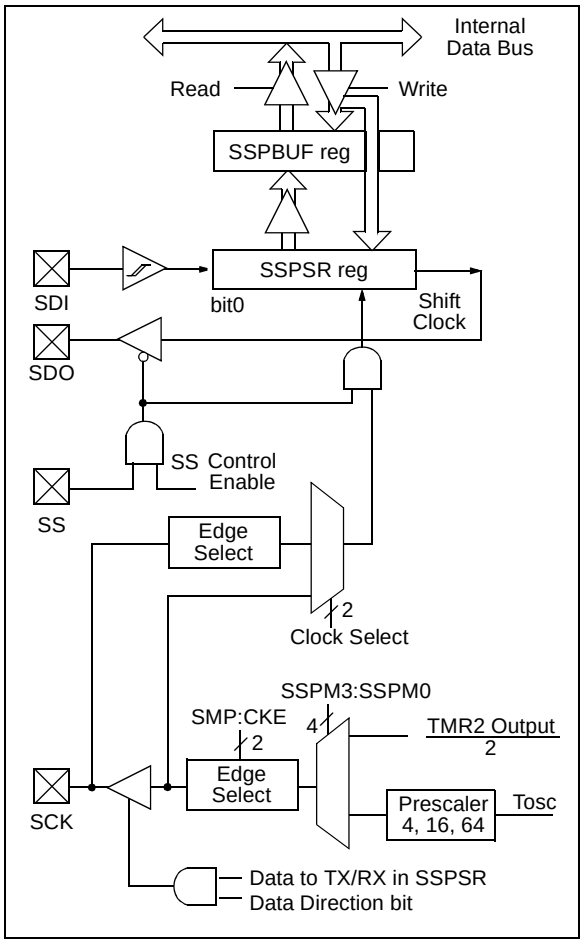


FIGURE 15-2: I^2C SLAVE MODE BLOCK DIAGRAM

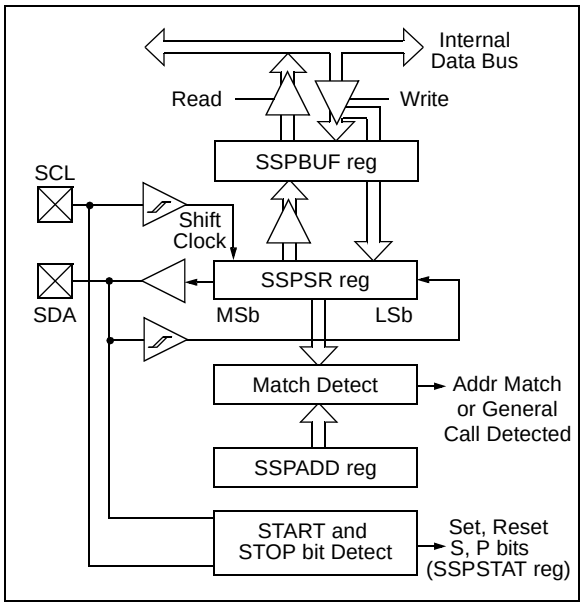
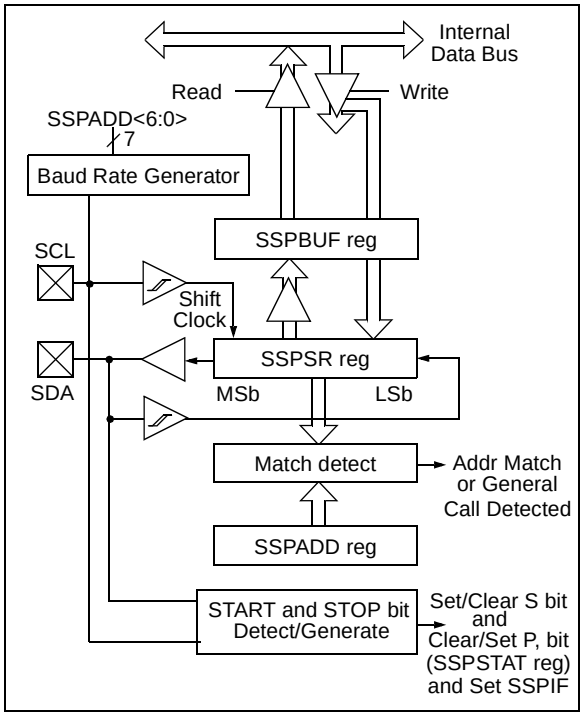


FIGURE 15-3: I^2C MASTER MODE BLOCK DIAGRAM



When the application software is expecting to receive valid data, the SSPBUF should be read before the next byte of data to transfer is written to the SSPBUF. Buffer full bit, BF (SSPSTAT<0>), indicates when SSPBUF has been loaded with the received data (transmission is complete). When the SSPBUF is read, bit BF is cleared. This data may be irrelevant if the SPI is only a transmitter. Generally the MSSP interrupt is used to determine when the transmission/reception has completed. The SSPBUF must be read and/or written. If the interrupt method is not going to be used, then software polling can be done to ensure that a write collision does not occur. Example 15-1 shows the loading of the SSPBUF (SSPSR) for data transmission.

EXAMPLE 15-1: LOADING THE SSPBUF (SSPSR) REGISTER

```
     MOVLB 6                ; Bank 6
LOOP BTFSS SSPSTAT, BF      ; Has data been
                            ;   received
                            ;   (transmit
                            ;   complete)?
     GOTO  LOOP             ; No
     MOVPF SSPBUF, RXDATA   ; Save in user RAM
     MOVFP TXDATA, SSPBUF   ; New data to xmit
```

The SSPSR is not directly readable, or writable and can only be accessed by addressing the SSPBUF register. Additionally, the MSSP status register (SSPSTAT) indicates the various status conditions.

15.1.2 ENABLING SPI I/O

To enable the serial port, MSSP Enable bit, SSPEN (SSPCON1<5>), must be set. To reset or reconfigure SPI mode, clear bit SSPEN, re-initialize the SSPCON registers and then set bit SSPEN. This configures the SDI, SDO, SCK and $\overline{SS}$ pins as serial port pins. For the pins to behave as the serial port function, some must have their data direction bits (in the DDR register) appropriately programmed. That is:

- SDI is automatically controlled by the SPI module
- SDO must have DDRB<7> cleared
- SCK (Master mode) must have DDRB<6> cleared
- SCK (Slave mode) must have DDRB<6> set
- $\overline{SS}$ must have PORTA<2> set

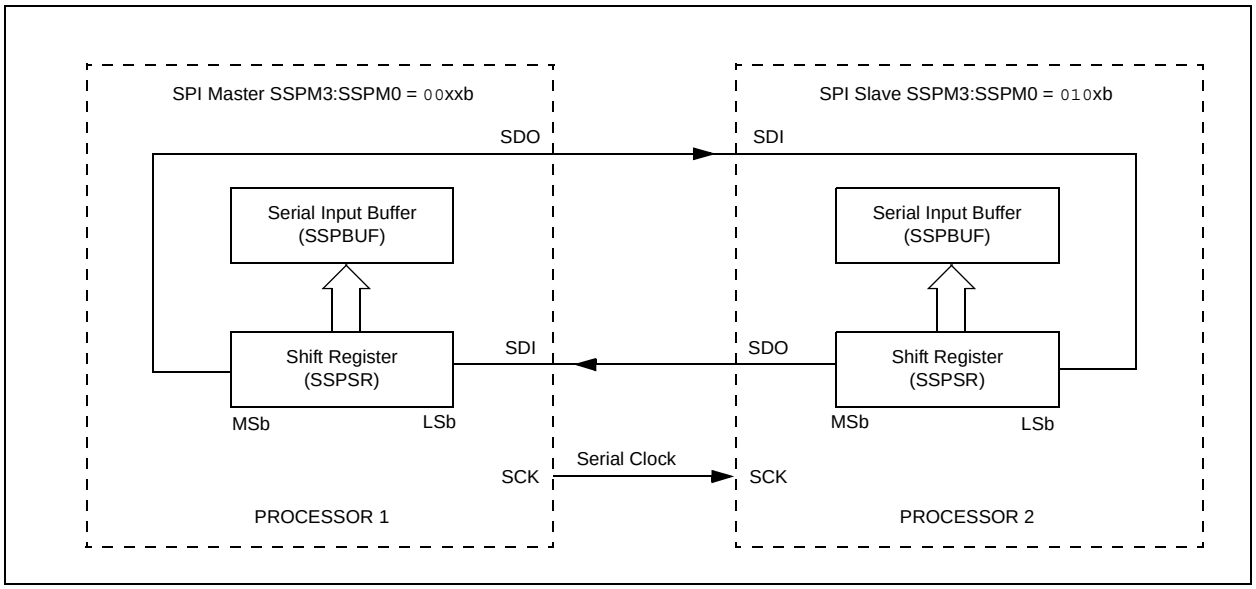
Any serial port function that is not desired may be overridden by programming the corresponding data direction (DDR) register to the opposite value.

15.1.3 TYPICAL CONNECTION

Figure 15-5 shows a typical connection between two microcontrollers. The master controller (Processor 1) initiates the data transfer by sending the SCK signal. Data is shifted out of both shift registers on their programmed clock edge and latched on the opposite edge of the clock. Both processors should be programmed to same Clock Polarity (CKP), then both controllers would send and receive data at the same time. Whether the data is meaningful (or dummy data) depends on the application software. This leads to three scenarios for data transmission:

- Master sends data — Slave sends dummy data
- Master sends data — Slave sends data
- Master sends dummy data — Slave sends data

FIGURE 15-5: SPI MASTER/SLAVE CONNECTION



15.2 MSSP I²C Operation

The MSSP module in I²C mode fully implements all master and slave functions (including general call support) and provides interrupts on START and STOP bits in hardware to determine a free bus (multi-master function). The MSSP module implements the standard mode specifications as well as 7-bit and 10-bit addressing.

Refer to Application Note AN578, *"Use of the SSP Module in the I²C Multi-Master Environment."*

A "glitch" filter is on the SCL and SDA pins when the pin is an input. This filter operates in both the 100 kHz and 400 kHz modes. In the 100 kHz mode, when these pins are an output, there is a slew rate control of the pin that is independent of device frequency.

FIGURE 15-10: I²C SLAVE MODE BLOCK DIAGRAM

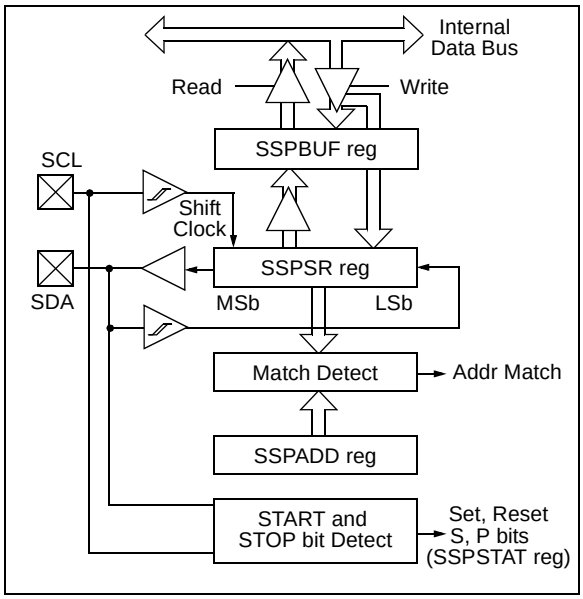
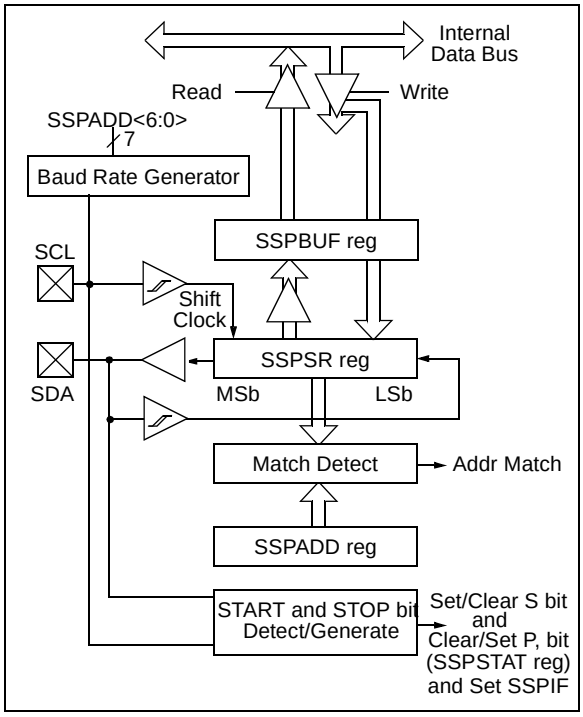


FIGURE 15-11: I²C MASTER MODE BLOCK DIAGRAM



Two pins are used for data transfer. These are the SCL pin, which is the clock and the SDA pin, which is the data. The SDA and SCL pins are automatically configured when the I²C mode is enabled. The SSP module functions are enabled by setting SSP Enable bit SSPEN (SSPCON1<5>).

The MSSP module has six registers for I²C operation. These are the:

- SSP Control Register1 (SSPCON1)
- SSP Control Register2 (SSPCON2)
- SSP Status Register (SSPSTAT)
- Serial Receive/Transmit Buffer (SSPBUF)
- SSP Shift Register (SSPSR) - Not directly accessible
- SSP Address Register (SSPADD)

The SSPCON1 register allows control of the I²C operation. Four mode selection bits (SSPCON1<3:0>) allow one of the following I²C modes to be selected:

- I²C Slave mode (7-bit address)
- I²C Slave mode (10-bit address)
- I²C Master mode, clock = OSC/4 (SSPADD +1)

Before selecting any I²C mode, the SCL and SDA pins must be programmed to inputs by setting the appropriate DDR bits. Selecting an I²C mode, by setting the SSPEN bit, enables the SCL and SDA pins to be used as the clock and data lines in I²C mode.

FIGURE 15-14: I²C SLAVE-TRANSMITTER (10-BIT ADDRESS)

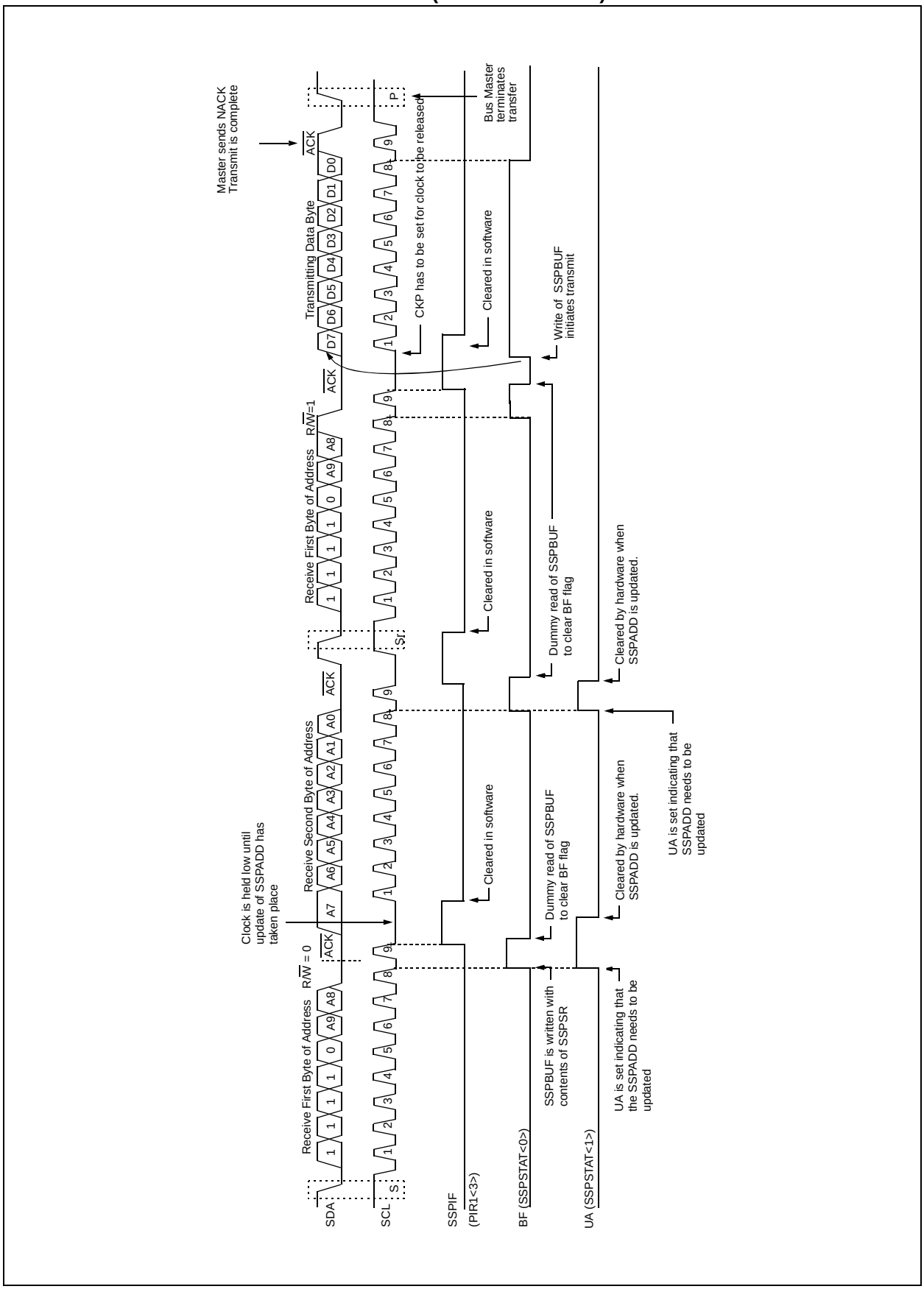
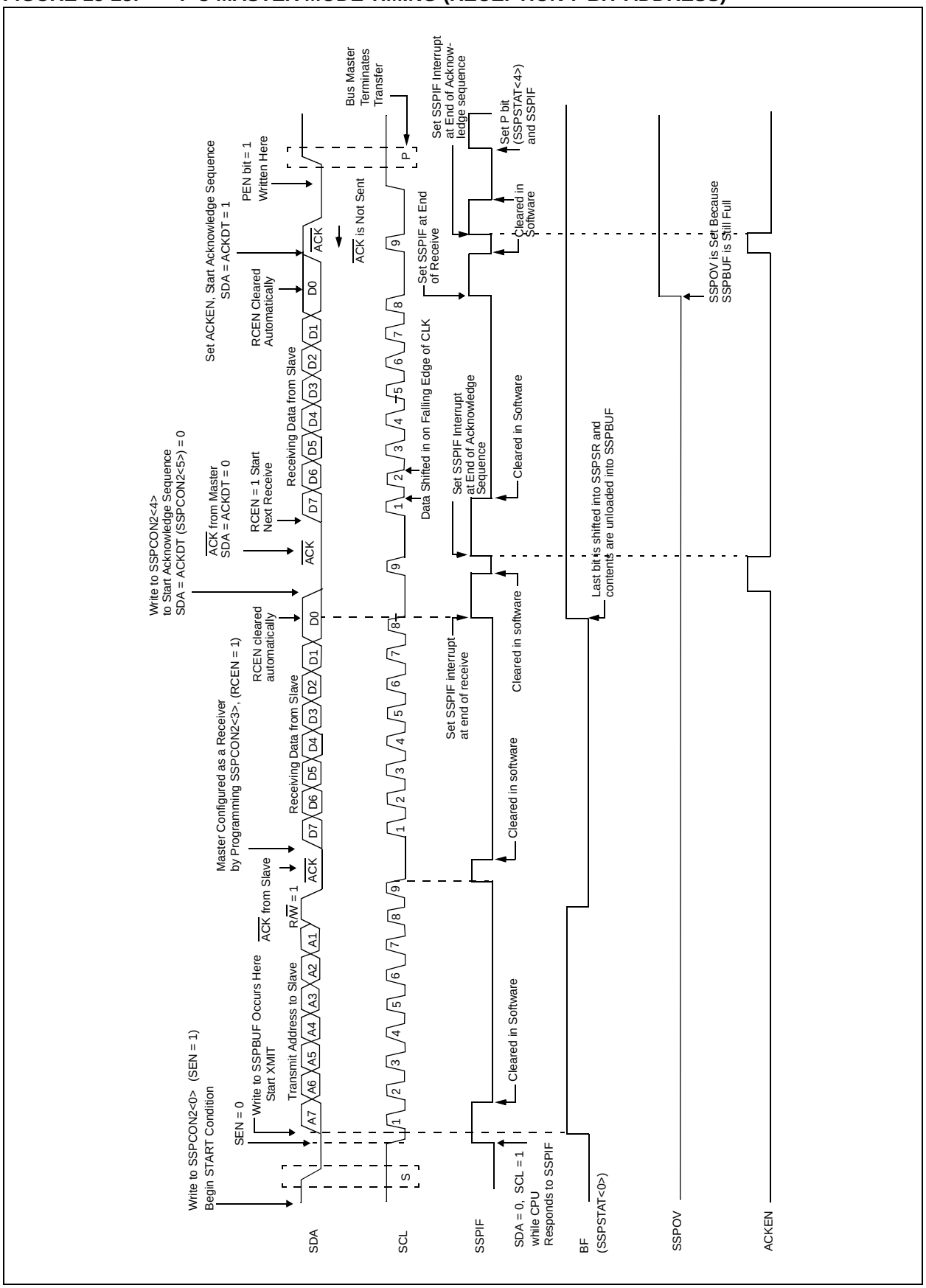


FIGURE 15-28: I²C MASTER MODE TIMING (RECEPTION 7-BIT ADDRESS)



17.3 Watchdog Timer (WDT)

The Watchdog Timer's function is to recover from software malfunction, or to reset the device while in SLEEP mode. The WDT uses an internal free running on-chip RC oscillator for its clock source. This does not require any external components. This RC oscillator is separate from the RC oscillator of the OSC1/CLKIN pin. That means that the WDT will run even if the clock on the OSC1/CLKIN and OSC2/CLKOUT pins has been stopped, for example, by execution of a SLEEP instruction. During normal operation, a WDT time-out generates a device RESET. The WDT can be permanently disabled by programming the configuration bits WDTPS1:WDTPS0 as '00' (Section 17.1).

Under normal operation, the WDT must be cleared on a regular interval. This time must be less than the minimum WDT overflow time. Not clearing the WDT in this time frame will cause the WDT to overflow and reset the device.

17.3.1 WDT PERIOD

The WDT has a nominal time-out period of 12 ms (with postscaler = 1). The time-out periods vary with temperature, VDD and process variations from part to part (see DC specs). If longer time-out periods are desired, configuration bits should be used to enable the WDT with a greater prescale. Thus, typical time-out periods up to 3.0 seconds can be realized.

The CLRWDT and SLEEP instructions clear the WDT and its postscale setting and prevent it from timing out, thus generating a device RESET condition.

The $\overline{TO}$ bit in the CPUSTA register will be cleared upon a WDT time-out.

17.3.2 CLEARING THE WDT AND POSTSCALER

The WDT and postscaler are cleared when:

- The device is in the RESET state
- A SLEEP instruction is executed
- A CLRWDT instruction is executed
- Wake-up from SLEEP by an interrupt

The WDT counter/postscaler will start counting on the first edge after the device exits the RESET state.

17.3.3 WDT PROGRAMMING CONSIDERATIONS

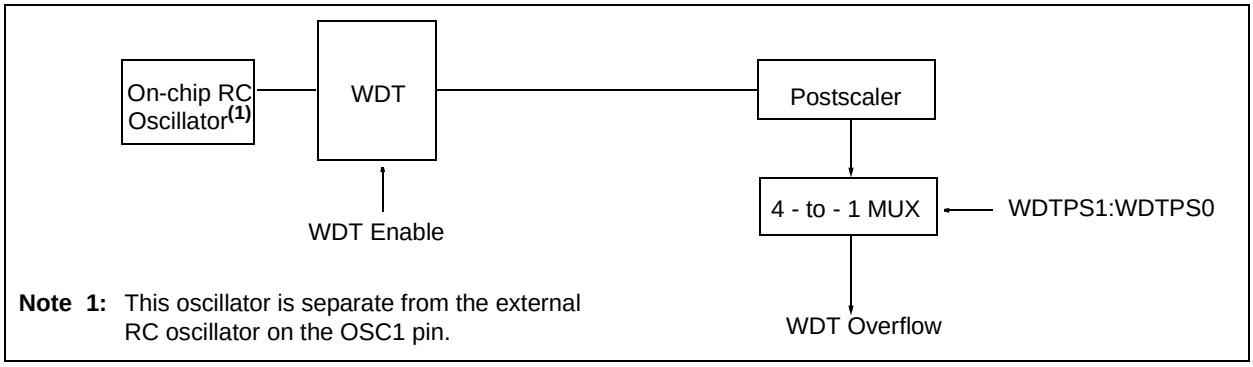
It should also be taken in account that under worst case conditions (VDD = Min., Temperature = Max., Max. WDT postscaler), it may take several seconds before a WDT time-out occurs.

The WDT and postscaler become the Power-up Timer whenever the PWRT is invoked.

17.3.4 WDT AS NORMAL TIMER

When the WDT is selected as a normal timer, the clock source is the device clock. Neither the WDT nor the postscaler are directly readable or writable. The overflow time is 65536 TOSC cycles. On overflow, the $\overline{TO}$ bit is cleared (device is not RESET). The CLRWDT instruction can be used to set the $\overline{TO}$ bit. This allows the WDT to be a simple overflow timer. The simple timer does not increment when in SLEEP.

FIGURE 17-1: WATCHDOG TIMER BLOCK DIAGRAM



Note 1: This oscillator is separate from the external RC oscillator on the OSC1 pin.

TABLE 17-2: REGISTERS/BITS ASSOCIATED WITH THE WATCHDOG TIMER

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	$\overline{MCLR}$, WDT
—	Config	See Figure 17-1 for location of WDTPSx bits in Configuration Word.								(Note 1)	(Note 1)
06h, Unbanked	CPUSTA	—	—	STKAV	GLINTD	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	--11 11qq	--11 qquu

Legend: - = unimplemented, read as '0', q = value depends on condition. Shaded cells are not used by the WDT.
Note 1: This value will be as the device was programmed, or if unprogrammed, will read as all '1's.

TABLE 18-2: PIC17CXXX INSTRUCTION SET (CONTINUED)

Mnemonic, Operands		Description	Cycles	16-bit Opcode				Status Affected	Notes
				MSb			LSb		
TSTFSZ	**f**	Test f, skip if 0	1 (2)	0011	0011	ffff	ffff	None	6,8
XORWF	**f,d**	Exclusive OR WREG with f	1	0000	110d	ffff	ffff	Z	
BIT-ORIENTED FILE REGISTER OPERATIONS									
BCF	**f,b**	Bit Clear f	1	1000	1bbb	ffff	ffff	None	
BSF	**f,b**	Bit Set f	1	1000	0bbb	ffff	ffff	None	
BTFSC	**f,b**	Bit test, skip if clear	1 (2)	1001	1bbb	ffff	ffff	None	6,8
BTFSS	**f,b**	Bit test, skip if set	1 (2)	1001	0bbb	ffff	ffff	None	6,8
BTG	**f,b**	Bit Toggle f	1	0011	1bbb	ffff	ffff	None	
LITERAL AND CONTROL OPERATIONS									
ADDLW	**k**	ADD literal to WREG	1	1011	0001	kkkk	kkkk	OV,C,DC,Z	
ANDLW	**k**	AND literal with WREG	1	1011	0101	kkkk	kkkk	Z	
CALL	**k**	Subroutine Call	2	111k	kkkk	kkkk	kkkk	None	7
CLRWDT	**—**	Clear Watchdog Timer	1	0000	0000	0000	0100	$\overline{TO}$, $\overline{PD}$	
GOTO	**k**	Unconditional Branch	2	110k	kkkk	kkkk	kkkk	None	7
IORLW	**k**	Inclusive OR literal with WREG	1	1011	0011	kkkk	kkkk	Z	
LCALL	**k**	Long Call	2	1011	0111	kkkk	kkkk	None	4,7
MOVLB	**k**	Move literal to low nibble in BSR	1	1011	1000	uuuu	kkkk	None	
MOVLR	**k**	Move literal to high nibble in BSR	1	1011	101x	kkkk	uuuu	None	
MOVLW	**k**	Move literal to WREG	1	1011	0000	kkkk	kkkk	None	
MULLW	**k**	Multiply literal with WREG	1	1011	1100	kkkk	kkkk	None	
RETFIE	**—**	Return from interrupt (and enable interrupts)	2	0000	0000	0000	0101	GLINTD	7
RETLW	**k**	Return literal to WREG	2	1011	0110	kkkk	kkkk	None	7
RETURN	**—**	Return from subroutine	2	0000	0000	0000	0010	None	7
SLEEP	**—**	Enter SLEEP mode	1	0000	0000	0000	0011	$\overline{TO}$, $\overline{PD}$	
SUBLW	**k**	Subtract WREG from literal	1	1011	0010	kkkk	kkkk	OV,C,DC,Z	
XORLW	**k**	Exclusive OR literal with WREG	1	1011	0100	kkkk	kkkk	Z	

Legend: Refer to Table 18-1 for opcode field descriptions.

Note 1: 2's Complement method.

2: Unsigned arithmetic.

3: If s = '1', only the file is affected: If s = '0', both the WREG register and the file are affected; If only the Working register (WREG) is required to be affected, then f = WREG must be specified.

4: During an LCALL, the contents of PCLATH are loaded into the MSB of the PC and kkkk kkkk is loaded into the LSB of the PC (PCL).

5: Multiple cycle instruction for EPROM programming when table pointer selects internal EPROM. The instruction is terminated by an interrupt event. When writing to external program memory, it is a two-cycle instruction.

6: Two-cycle instruction when condition is true, else single cycle instruction.

7: Two-cycle instruction except for TABLRD to PCL (program counter low byte), in which case it takes 3 cycles.

8: A "skip" means that instruction fetched during execution of current instruction is not executed, instead a NOP is executed.

MULLW **Multiply Literal with WREG**

Syntax: [*label*] MULLW k

Operands: $0 \le k \le 255$

Operation: (k x WREG) → PRODH:PRODL

Status Affected: None

Encoding:

1011	1100	kkkk	kkkk

Description: An unsigned multiplication is carried out between the contents of WREG and the 8-bit literal 'k'. The 16-bit result is placed in PRODH:PRODL register pair. PRODH contains the high byte.

WREG is unchanged.

None of the status flags are affected.

Note that neither overflow, nor carry is possible in this operation. A zero result is possible, but not detected.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write registers PRODH: PRODL

Example: `MULLW 0xC4`

Before Instruction
- WREG = 0xE2
- PRODH = ?
- PRODL = ?

After Instruction
- WREG = 0xC4
- PRODH = 0xAD
- PRODL = 0x08

MULWF **Multiply WREG with f**

Syntax: [*label*] MULWF f

Operands: $0 \le f \le 255$

Operation: (WREG x f) → PRODH:PRODL

Status Affected: None

Encoding:

0011	0100	ffff	ffff

Description: An unsigned multiplication is carried out between the contents of WREG and the register file location 'f'. The 16-bit result is stored in the PRODH:PRODL register pair. PRODH contains the high byte.

Both WREG and 'f' are unchanged.

None of the status flags are affected.

Note that neither overflow, nor carry is possible in this operation. A zero result is possible, but not detected.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write registers PRODH: PRODL

Example: `MULWF REG`

Before Instruction
- WREG = 0xC4
- REG = 0xB5
- PRODH = ?
- PRODL = ?

After Instruction
- WREG = 0xC4
- REG = 0xB5
- PRODH = 0x8A
- PRODL = 0x94

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

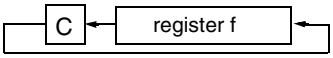
Operation: f<n> → d<n+1>;
f<7> → C;
C → d<0>

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

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.

NOTES: