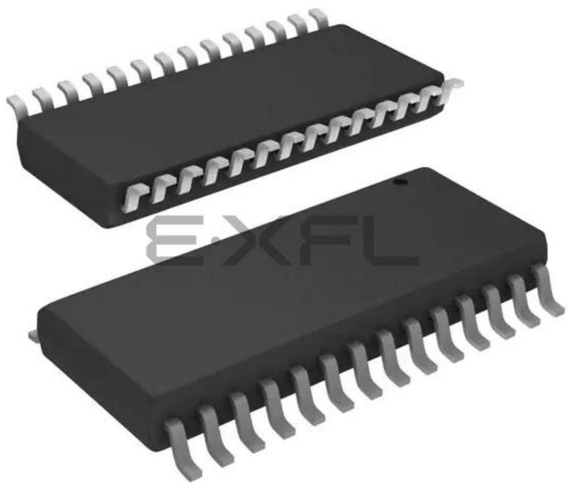
Microchip Technology - PIC16C642-04I/SO Datasheet



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	4MHz
Connectivity	-
Peripherals	Brown-out Detect/Reset, LED, POR, WDT
Number of I/O	22
Program Memory Size	7KB (4K x 14)
Program Memory Type	OTP
EEPROM Size	-
RAM Size	176 x 8
Voltage - Supply (Vcc/Vdd)	4V ~ 6V
Data Converters	-
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-SOIC (0.295", 7.50mm Width)
Supplier Device Package	28-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic16c642-04i-so

1.0 GENERAL DESCRIPTION

PIC16C64X & PIC16C66X devices are 28-pin and 40-pin EPROM-based members of the versatile PIC16CXXX family of low-cost, high-performance, CMOS, fully-static, 8-bit microcontrollers.

All PIC16/17 microcontrollers employ an advanced RISC architecture. The PIC16CXXX family has enhanced core features, eight-level deep stack, and multiple internal and external interrupt sources. The separate instruction and data buses of the Harvard architecture allow a 14-bit wide instruction word with the separate 8-bit wide data. The two-stage instruction pipeline allows all instructions to execute in a single-cycle, except for program branches (which require two cycles). A total of 35 instructions (reduced instruction set) are available. Additionally, a large register set gives some of the architectural innovations used to achieve a very high performance.

PIC16CXXX microcontrollers typically achieve a 2:1 code compression and a 4:1 speed improvement over other 8-bit microcontrollers in its class.

The PIC16C641 has 128 bytes of RAM and the PIC16C642 has 176 bytes of RAM. Both devices have 22 I/O pins, and an 8-bit timer/counter with an 8-bit programmable prescaler. In addition, they have two analog comparators with a programmable on-chip voltage reference module. Program Memory has internal parity error detection circuitry with a Parity Error Reset. The comparator module is ideally suited for applications requiring a low-cost analog interface (e.g., battery chargers, threshold detectors, white goods controllers, etc.).

The PIC16C661 has 128 bytes of RAM and the PIC16C662 has 176 bytes of RAM. Both devices have 33 I/O pins, and an 8-bit timer/counter with an 8-bit programmable prescaler. They also have an 8-bit Parallel Slave Port. In addition, the devices have two analog comparators with a programmable on-chip voltage reference module. Program Memory has internal parity error detection circuitry with a Parity Error Reset. The comparator module is ideally suited for applications requiring a low-cost analog interface (e.g., battery chargers, threshold detectors, white goods controllers, etc.).

PIC16CXXX devices have special features to reduce external components, thus reducing cost, enhancing system reliability and reducing power consumption. There are four oscillator options, of which the single pin RC oscillator provides a low-cost solution, the LP oscillator minimizes power consumption, XT is a standard crystal, and the HS is for High Speed crystals. The SLEEP (power-down) mode offers power saving. The user can wake-up the chip from SLEEP through several external and internal interrupts and resets.

A highly reliable Watchdog Timer (WDT) with its own on-chip RC oscillator provides protection against software lock-up.

A UV-erasable CERDIP-packaged version is ideal for code development while the cost-effective One-Time Programmable (OTP) version is suitable for production in any volume.

The PIC16CXXX series fit perfectly in applications ranging from battery chargers to low-power remote sensors. The EPROM technology makes customization of application programs (detection levels, pulse generation, timers, etc.) extremely fast and convenient. The small footprint packages make this microcontroller series perfect for all applications with space limitations. Low-cost, low-power, high-performance, ease of use, and I/O flexibility make the PIC16C64X & PIC16C66X very versatile.

1.1 Family and Upward Compatibility

Those users familiar with the PIC16C5X family of microcontrollers will realize that this is an enhanced version of the PIC16C5X architecture. Please refer to Appendix A for a detailed list of enhancements. Code written for PIC16C5X can be easily ported to the PIC16C64X & PIC16C66X (Appendix B).

1.2 Development Support

PIC16C64X & PIC16C66X devices are supported by the complete line of Microchip Development tools, including:

- MPLAB Integrated Development Environment including MPLAB-Simulator.
- MPASM Universal Assembler and MPLAB-C Universal C compiler.
- PRO MATE II and PICSTART Plus device programmers.
- PICMASTER In-circuit Emulator System
- *fuzzy*TECH-MP Fuzzy Logic Development Tools
- DriveWay Visual Programming Tool

Please refer to Section 11.0 for more details about these and other Microchip development tools.

TABLE 3-2: PIC16C661/662 PINOUT DESCRIPTION

Name	DIP Pin #	QFP Pin #	PLCC Pin #	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	13	30	14	I	ST/CMOS	Oscillator crystal input or external clock source input.
OSC2/CLKOUT	14	31	15	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
$\overline{\text{MCLR}}$/VPP	1	18	2	I/P	ST	Master clear (reset) input or programming voltage input. This pin is an active low reset to the device.
						PORTA is a bi-directional I/O port.
RA0/AN0	2	19	3	I/O	ST	Analog comparator input.
RA1/AN1	3	20	4	I/O	ST	Analog comparator input.
RA2/AN2/VREF	4	21	5	I/O	ST	Analog comparator input or VREF output.
RA3/AN3	5	22	6	I/O	ST	Analog comparator input or comparator output.
RA4/T0CKI	6	23	7	I/O	ST	Can be selected to be the clock input to the Timer0 timer/counter or a comparator output. Output is open drain type.
RA5	7	24	8	I/O	ST	
						PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs.
RB0/INT	33	8	36	I/O	TTL/ST(1)	RB0 can also be selected as an external interrupt pin.
RB1	34	9	37	I/O	TTL	
RB2	35	10	38	I/O	TTL	
RB3	36	11	39	I/O	TTL	
RB4	37	14	41	I/O	TTL	Interrupt on change pin.
RB5	38	15	42	I/O	TTL	Interrupt on change pin.
RB6	39	16	43	I/O	TTL/ST(2)	Interrupt on change pin. Serial programming clock.
RB7	40	17	44	I/O	TTL/ST(2)	Interrupt on change pin. Serial programming data.
						PORTC is a bi-directional I/O port.
RC0	15	32	16	I/O	ST	
RC1	16	35	18	I/O	ST	
RC2	17	36	19	I/O	ST	
RC3	18	37	20	I/O	ST	
RC4	23	42	25	I/O	ST	
RC5	24	43	26	I/O	ST	
RC6	25	44	27	I/O	ST	
RC7	26	1	29	I/O	ST	

Legend: O = output, I/O = input/output, P = power, I = input, — = not used, ST = Schmitt Trigger input, TTL = TTL input

Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
2: This buffer is a Schmitt Trigger input when used in serial programming mode.
3: This buffer is a Schmitt Trigger input when configured as a general purpose I/O and a TTL input when used in the Parallel Slave Port Mode (for interfacing to a microprocessor port).

4.2.2.1 STATUS REGISTER

The STATUS register, shown in Figure 4-5, contains the arithmetic status of the ALU, the RESET status, and the bank select bits for data memory.

The STATUS register can be the destination for any instruction, like any other register. If the STATUS register is the destination for an instruction that affects the Z, DC or C bits, then the write to these three bits is disabled. These bits are set or cleared according to the device logic. Furthermore, the $\overline{\text{TO}}$ and $\overline{\text{PD}}$ bits are not writable. Therefore, the result of an instruction with the STATUS register as destination may be different than intended.

For example, `CLRF STATUS` will clear the upper-three bits and set the Z bit. This leaves the STATUS register as `000uu1uu` (where u = unchanged).

It is recommended, therefore, that only `BCF`, `BSF`, `SWAPF`, and `MOVWF` instructions are used to alter the STATUS register because these instructions do not affect any status bit. For other instructions, not affecting any status bits, see the “Instruction Set Summary.”

Note 1: The IRP and RP1 bits (STATUS<7:6>) are reserved on the PIC16C64X & PIC16C66X and should be maintained clear. Use of these bits as general purpose R/W bits is NOT recommended, since this may affect upward compatibility with future products.

Note 2: The C and DC bits operate as a $\overline{\text{Borrow}}$ and $\overline{\text{Digit Borrow}}$ out bit, respectively, in subtraction. See the `SUBLW` and `SUBWF` instructions for examples.

FIGURE 4-5: STATUS REGISTER (ADDRESS 03h, 83h)

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{\text{TO}}$	$\overline{\text{PD}}$	Z	DC	C
bit7							bit0

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as ‘0’
- n = Value at POR reset

bit 7: **IRP:** Register Bank Select bit (used for indirect addressing)
1 = Bank 2, 3 (100h - 1FFh)
0 = Bank 0, 1 (00h - FFh)
Bit IRP is reserved on the PIC16C64X & PIC16C66X, always maintain this bit clear.

bit 6-5: **RP1:RP0**: Register Bank Select bits (used for direct addressing)
`11` = Bank 3 (180h - 1FFh)
`10` = Bank 2 (100h - 17Fh)
`01` = Bank 1 (80h - FFh)
`00` = Bank 0 (00h - 7Fh)
Each bank is 128 bytes. Bit RP1 is reserved on the PIC16C64X & PIC16C66X, always maintain this bit clear.

bit 4: **$\overline{\text{TO}}$**: Time-out bit
1 = After power-up, `CLRWDT` instruction, or `SLEEP` instruction
0 = A WDT time-out occurred

bit 3: **$\overline{\text{PD}}$**: Power-down bit
1 = After power-up or by the `CLRWDT` instruction
0 = By execution of the `SLEEP` instruction

bit 2: **Z**: Zero bit
1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

bit 1: **DC**: Digit carry/$\overline{\text{borrow}}$ bit (`ADDWF`, `ADDLW`,`SUBLW`,`SUBWF` instructions) (for $\overline{\text{borrow}}$ the polarity is reversed)
1 = A carry-out from the 4th low order bit of the result occurred
0 = No carry-out from the 4th low order bit of the result

bit 0: **C**: Carry/$\overline{\text{borrow}}$ bit (`ADDWF`, `ADDLW`,`SUBLW`,`SUBWF` instructions)
1 = A carry-out from the most significant bit of the result occurred
0 = No carry-out from the most significant bit of the result occurred
Note: For $\overline{\text{borrow}}$ the polarity is reversed. A subtraction is executed by adding the two’s complement of the second operand. For rotate (`RRF`, `RLF`) instructions, this bit is loaded with either the high or low order bit of the source register.

4.2.2.2 OPTION REGISTER

The OPTION register is a readable and writable register which contains various control bits to configure the TMR0/WDT prescaler, the external RB0/INT interrupt, TMR0, and the weak pull-ups on PORTB.

Note:	To achieve a 1:1 prescaler assignment for TMR0, assign the prescaler to the WDT.

FIGURE 4-6: OPTION REGISTER (ADDRESS 81h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit7							bit0

R= Readable bit
W= Writable bit
U= Unimplemented bit, read as ‘0’
- n= Value at POR reset

bit 7: **$\overline{\text{RBPU}}$**: PORTB Pull-up Enable bit
1 = PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled by individual port latch values

bit 6: **INTEDG**: Interrupt Edge Select bit
1 = Interrupt on rising edge of RB0/INT pin
0 = Interrupt on falling edge of RB0/INT pin

bit 5: **T0CS**: TMR0 Clock Source Select bit
1 = Transition on RA4/T0CKI pin
0 = Internal instruction cycle clock (CLKOUT)

bit 4: **T0SE**: TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on RA4/T0CKI pin
0 = Increment on low-to-high transition on RA4/T0CKI pin

bit 3: **PSA**: Prescaler Assignment bit
1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

bit 2-0: **PS2:PS0**: Prescaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

5.7 Parallel Slave Port (PIC16C661 and PIC16C662 only)

PORTD operates as an 8-bit wide parallel slave port, or as a microprocessor port when control bit PSPMODE (TRISE<4>) is set. In slave mode it is asynchronously readable and writable by the external world through $\overline{\text{RD}}$ control input pin (RE0/$\overline{\text{RD}}$) and $\overline{\text{WR}}$ control input pin (RE1/$\overline{\text{WR}}$).

It can directly interface to an 8-bit microprocessor data bus. The external microprocessor can read or write the PORTD latch as an 8-bit latch. Setting PSPMODE enables port pin RE0/$\overline{\text{RD}}$ to be the $\overline{\text{RD}}$ input, RE1/$\overline{\text{WR}}$ to be the $\overline{\text{WR}}$ input and RE2/$\overline{\text{CS}}$ to be the $\overline{\text{CS}}$ (chip select) input. For this functionality, the corresponding data direction bits of the TRISE register (TRISE<2:0>) must be configured as inputs (set).

There are actually two 8-bit latches, one for data-out (from the PIC16/17) and one for data input. The user writes 8-bit data to PORTD data latch and reads data from the port pin latch (note that they have the same address). In this mode, the TRISD register is ignored since the microprocessor is controlling the direction of data flow.

Input Buffer Full Status Flag bit IBF (TRISE<7>) is set if a received word is waiting to be read by the CPU. Once the PORTD input latch is read, bit IBF is cleared. IBF is a read only status bit. Output Buffer Full Status Flag bit OBF (TRISE<6>) is set if a word written to PORTD latch is waiting to be read by the external bus. Once the PORTD output latch is read by the microprocessor, bit OBF is cleared. Input Buffer Overflow Status flag bit IBOV (TRISE<5>) is set if a second write to the microprocessor port is attempted when the previous word has not been read by the CPU (the first word is retained in the buffer).

When not in Parallel Slave Port mode, bits IBF and OBF are held clear. However, if flag bit IBOV was previously set, it must be cleared in software.

An interrupt is generated and latched into flag bit PSPIF (PIR1<7>) when a read or a write operation is completed. Flag bit PSPIF must be cleared by user software. The interrupt can be disabled by clearing the interrupt enable bit PSPIE (PIE1<7>).

FIGURE 5-12: PORTD AND PORTE AS A PARALLEL SLAVE PORT

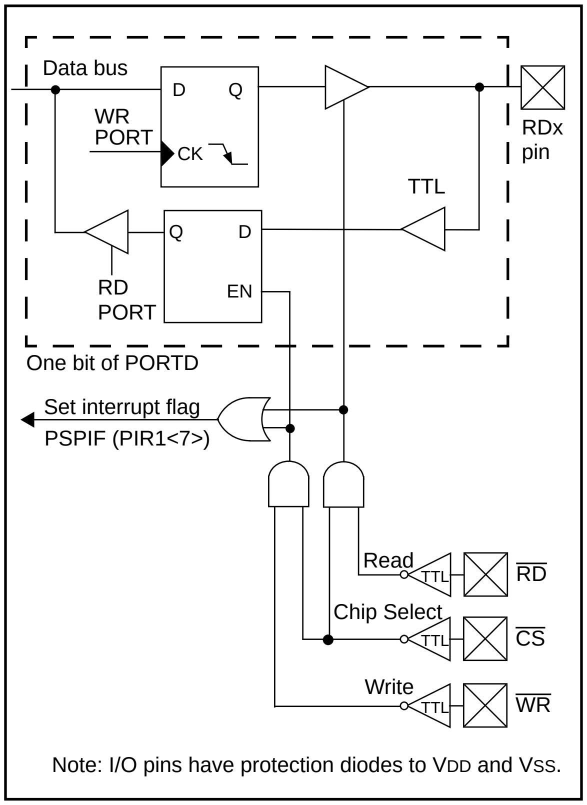


TABLE 5-11: REGISTERS ASSOCIATED WITH PARALLEL SLAVE PORT

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets
08h	PORTD	PSP7	PSP6	PSP5	PSP4	PSP3	PSP2	PSP1	PSP0	xxxx xxxx	uuuu uuuu
09h	PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -xxx	---- -uuu
89h	TRISE	IBF	OBF	IBOV	PSPMODE	—	TRISE2	TRISE1	TRISE0	0000 -111	0000 -111
0Ch	PIR1	PSPIF(1)	CMIF	—	—	—	—	—	—	00-- ----	00-- ----
8Ch	PIE1	PSPIE(1)	CMIE	—	—	—	—	—	—	00-- ----	00-- ----

Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'. Shaded cells are not used by the PSP.

Note 1: These bits are reserved on the PIC16C641/642, always maintain these bits clear.

6.2 Using Timer0 with External Clock

When an external clock input is used for Timer0, it must meet certain requirements. The requirements ensure the external clock can be synchronized with the internal phase clock (TOSC). Also, there is a delay in the actual incrementing of Timer0 after synchronization.

6.2.1 EXTERNAL CLOCK SYNCHRONIZATION

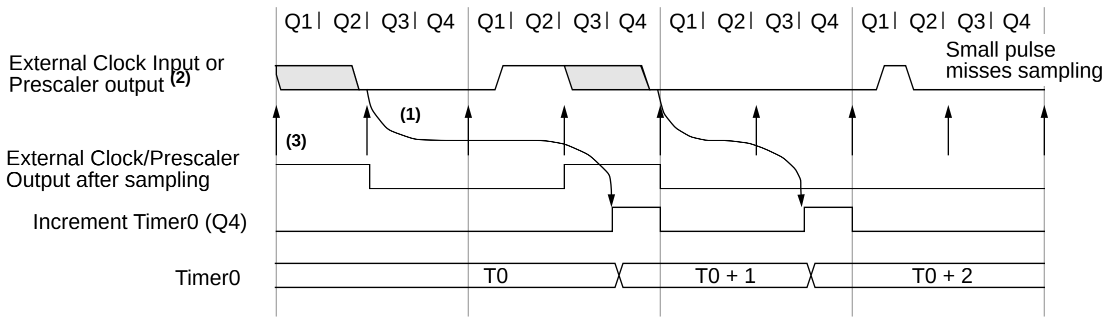
When no prescaler is used, the external clock input is the same as the prescaler output. The synchronization of T0CKI with the internal phase clocks is accomplished by sampling the prescaler output on the Q2 and Q4 cycles of the internal phase clocks (Figure 6-5). Therefore, it is necessary for T0CKI to be high for at least 2Tosc (and a small RC delay of 20 ns) and low for at least 2Tosc (and a small RC delay of 20 ns). Refer to the electrical specification of the desired device.

When a prescaler is used, the external clock input is divided by the asynchronous ripple-counter type prescaler so that the prescaler output is symmetrical. For the external clock to meet the sampling requirement, the ripple-counter must be taken into account. Therefore, it is necessary for T0CKI to have a period of at least 4Tosc (and a small RC delay of 40 ns) divided by the prescaler value. The only requirement on T0CKI high and low time is that they do not violate the minimum pulse width requirement of 10 ns. Refer to parameters 40, 41, and 42 in the electrical specification of the desired device.

6.2.2 TIMER0 INCREMENT DELAY

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 the Timer0 module is actually incremented. Figure 6-5 shows the delay from the external clock edge to the timer incrementing.

FIGURE 6-5: TIMER0 TIMING WITH EXTERNAL CLOCK


Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4
External Clock Input or Prescaler output (2)
Small pulse misses sampling
(1)
(3)
External Clock/Prescaler Output after sampling
Increment Timer0 (Q4)
Timer0
T0
T0 + 1
T0 + 2

Note 1: Delay from clock input change to Timer0 increment is 3Tosc to 7Tosc. (Duration of Q = Tosc). Therefore, the error in measuring the interval between two edges on Timer0 input = ±4Tosc max.
2: External clock if no prescaler selected, prescaler output otherwise.
3: The arrows indicate the points in time where sampling occurs.

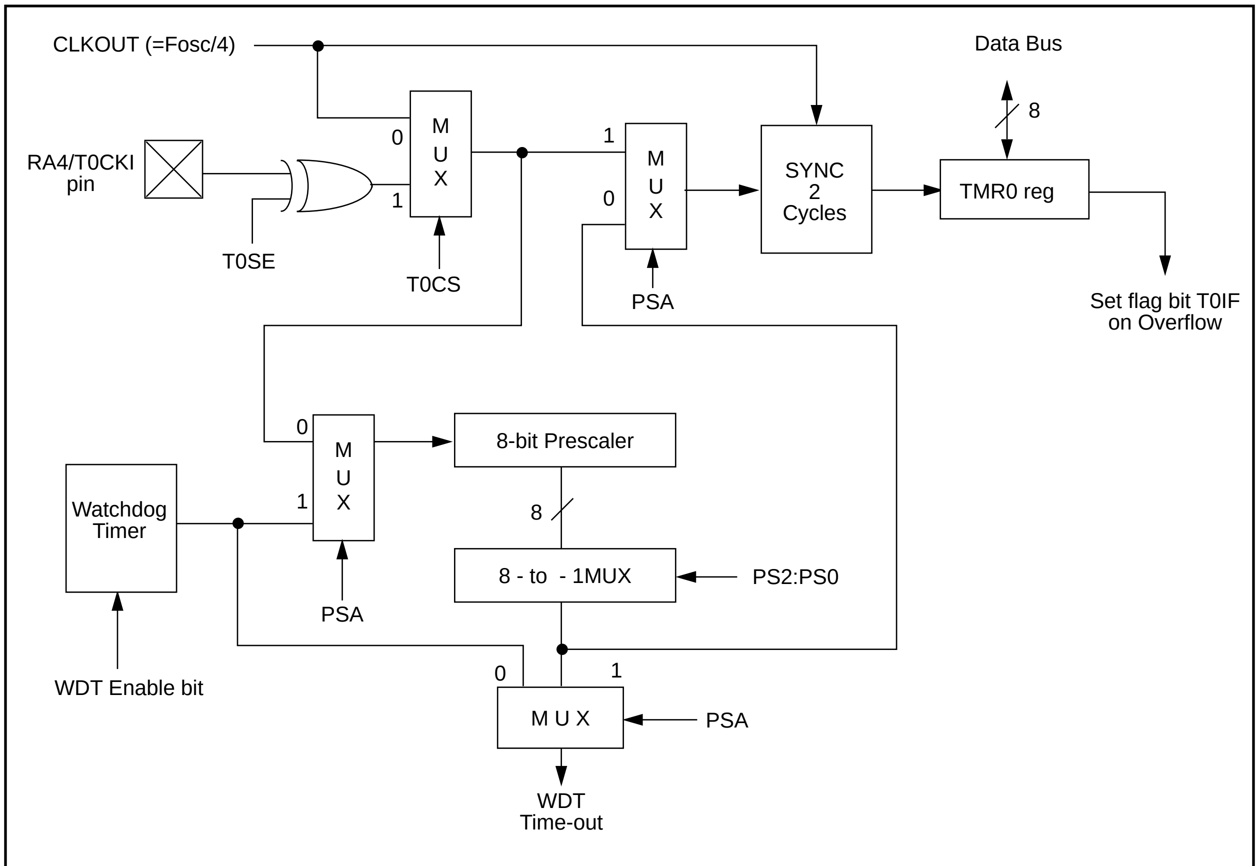
6.3 Prescaler

An 8-bit counter is available as a prescaler for the Timer0 module or as a postscaler for the Watchdog Timer (WDT), respectively (Figure 6-6). For simplicity, this counter is being referred to as "prescaler" throughout this data sheet. Note that the prescaler may be used by either the Timer0 module or the Watchdog Timer, but not both. Thus, a prescaler assignment for the Timer0 module means that there is no prescaler for the Watchdog Timer, and vice-versa.

The PSA and PS2:PS0 bits (OPTION<3:0>) determine the prescaler assignment and prescale ratio.

When assigned to the Timer0 module, all instructions writing to the TMR0 register (e.g., `CLRF 1, MOVWF 1, BSF 1,x`) will clear the prescaler count. When assigned to Watchdog Timer, a `CLRWDT` instruction will clear the prescaler count along with the Watchdog Timer. The prescaler is not readable or writable.

FIGURE 6-6: BLOCK DIAGRAM OF THE TIMER0/WDT PRESCALER



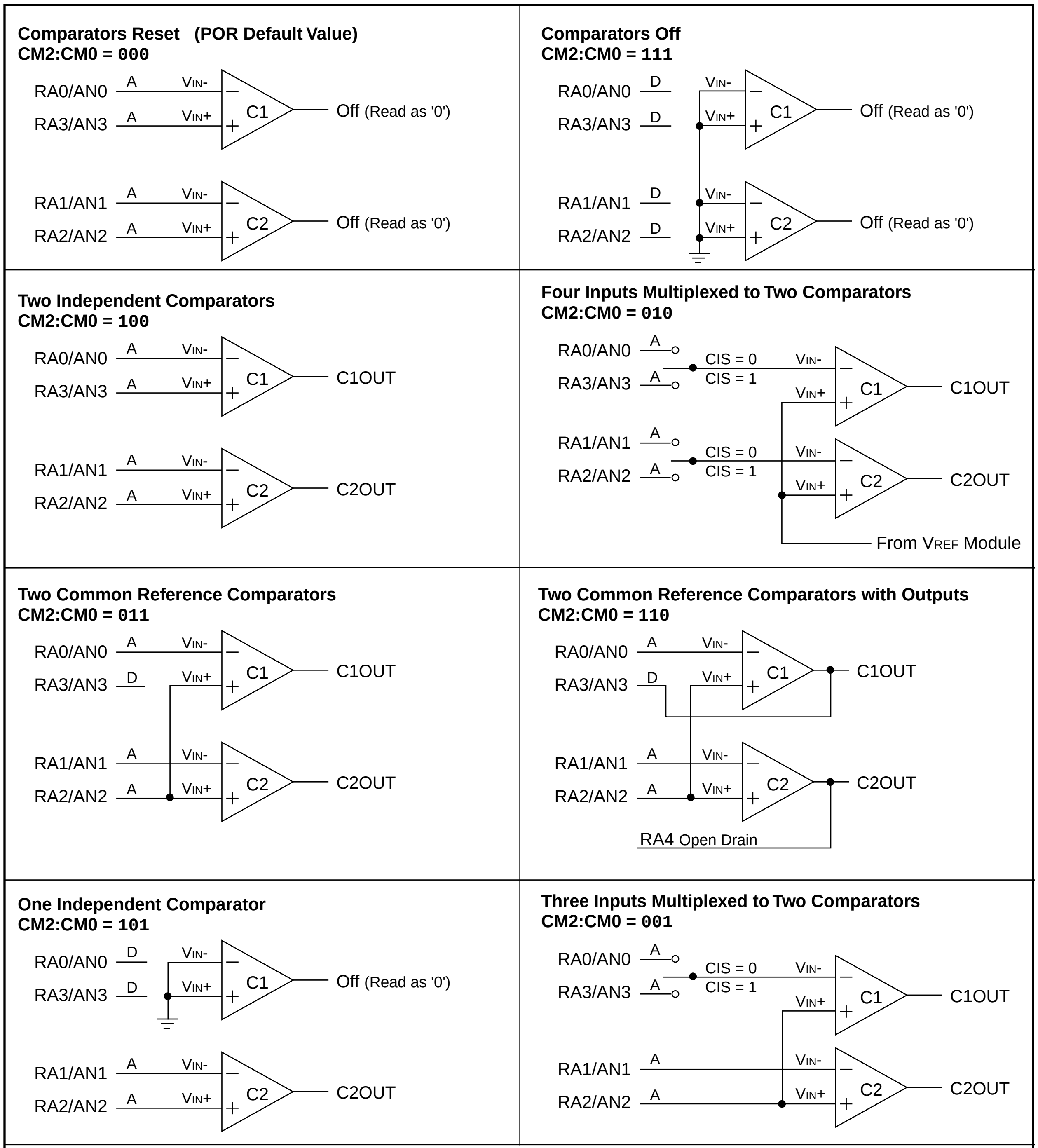
Note: T0CS, T0SE, PSA, PS2:PS0 are (OPTION<5:0>).

7.1 Comparator Configuration

There are eight modes of operation for the comparators. The CMCON register is used to select the mode. Figure 7-2 shows the eight possible modes. The TRISA register controls the data direction of the comparator pins for each mode. If the comparator mode is changed, the comparator output level may not be valid for the specified mode change delay shown in Table 12-2.

Note:	Comparator interrupts should be disabled during a comparator mode change otherwise a false interrupt may occur.

FIGURE 7-2: COMPARATOR I/O OPERATING MODES



A = Analog Input, port reads zeros always.
D = Digital Input.
CIS (CMCON<3>) is the Comparator Input Switch.

7.6 Comparator Interrupts

The comparator interrupt flag is set whenever there is a change in the output value of either comparator. User software will need to maintain information about the status of the output bits, as read from CMCON<7:6>, to determine the actual change that has occurred. The CMIF bit (PIR1<6>), is the comparator interrupt flag and must be cleared in user software.

To enable the Comparator interrupt the following bits must be set:

- CMIE (PIE1<6>)
- PEIE (INTCON<6>)
- GIE (INTCON<7>)

The user, in the interrupt service routine, can clear the interrupt in the following manner:

a) Any read or write of CMCON. This will end the mismatch condition.

b) Clear flag bit CMIF.

A mismatch condition will continue to set flag bit CMIF. Reading CMCON will end the mismatch condition, and allow flag bit CMIF to be cleared.

7.7 Comparator Operation During SLEEP

When a comparator is active and the device is placed in SLEEP mode, the comparator remains active and the interrupt is functional if enabled. This interrupt will wake up the device from SLEEP mode when enabled. While the comparator is powered up, higher sleep currents than shown in the power-down current specification will occur. Each comparator that is operational will consume additional current as shown in the comparator specifications. To minimize power consumption while in SLEEP mode, turn off the comparators, CM2:CM0 = 111, before entering sleep. If the device wakes up from sleep, the contents of the CMCON register are not affected.

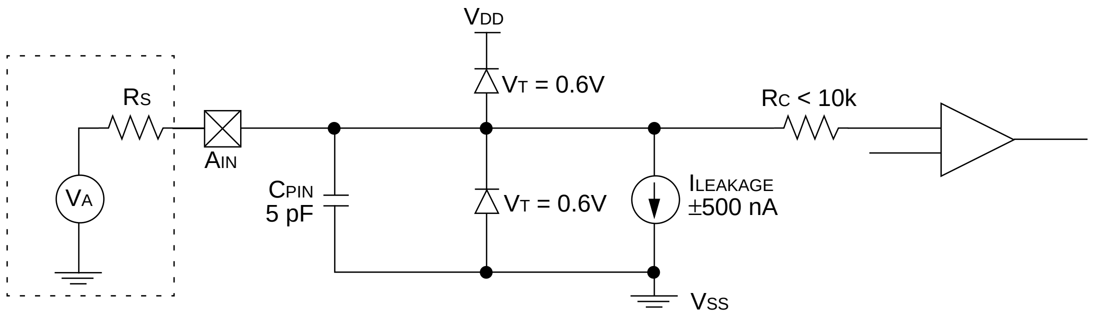
7.8 Effects of a RESET

A device reset forces the CMCON register to its reset state. This forces the comparator module to be in the comparator reset mode, CM2:CM0 = 000. This ensures that all potential inputs are analog inputs. Device current is minimized when analog inputs are present at reset time. The comparators will be powered down during the reset interval.

7.9 Analog Input Connection Considerations

A simplified circuit for an analog input is shown in Figure 7-5. Since the analog pins are connected to a digital output, they have reverse biased diodes to V_{DD} and V_{SS}. The analog input therefore, must be between V_{SS} and V_{DD}. If the input voltage deviates from this range by more than 0.6V in either direction, one of the diodes is forward biased and a latch-up may occur. A maximum source impedance of 10 kΩ is recommended for the analog sources. Any external component connected to an analog input pin, such as a capacitor or a Zener diode, should have very little leakage current.

FIGURE 7-5: ANALOG INPUT MODEL



Legend	C_{PIN}	= Input Capacitance
	V_T	= Threshold Voltage
	$I_{LEAKAGE}$	= Leakage Current at the pin due to various junctions
	R_{IC}	= Interconnect Resistance
	R_S	= Source Impedance
	V_A	= Analog Voltage

9.3 Reset

The PIC16CXXX differentiates between various kinds of reset:

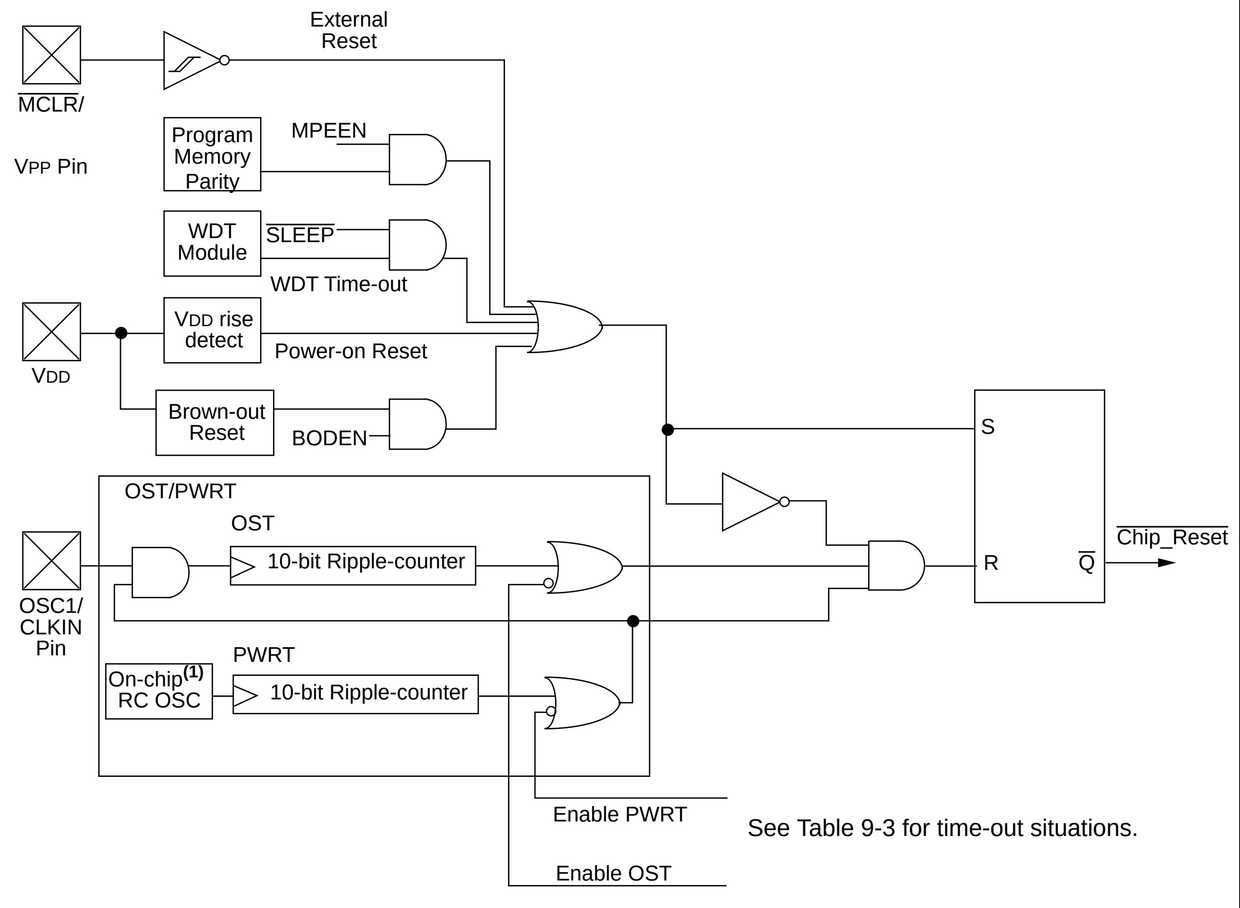
a) Power-on reset (POR)
b) $\overline{\text{MCLR}}$ reset during normal operation
c) $\overline{\text{MCLR}}$ reset during SLEEP
d) WDT reset (normal operation)
e) Brown-out Reset (BOR)
f) Parity Error Reset (PER)

Some registers are not affected in any reset condition; their status is unknown on POR and unchanged in any other reset. Most other registers are reset to a “reset state” on Power-on reset, $\overline{\text{MCLR}}$, WDT reset, Brown-out Reset, Parity Error Reset, and on $\overline{\text{MCLR}}$ reset during SLEEP. They are not affected by a WDT wake-up, since this is viewed as the resumption of normal operation. $\overline{\text{TO}}$ and $\overline{\text{PD}}$ bits are set or cleared differently in different reset situations as indicated in Table 9-4. These bits are used in software to determine the nature of the reset. See Table 9-6 for a full description of reset states of all registers.

A simplified block diagram of the on-chip reset circuit is shown in Figure 9-7.

The $\overline{\text{MCLR}}$ reset path has a noise filter to detect and ignore small pulses. See Table 12-6 for pulse width specification.

FIGURE 9-7: SIMPLIFIED BLOCK DIAGRAM OF ON-CHIP RESET CIRCUIT



See Table 9-3 for time-out situations.

Note 1: This is a separate oscillator from the RC oscillator of the CLKIN pin.

9.5 Interrupts

The PIC16C641 and PIC16C642 have four sources of interrupt, while the PIC16C661 and PIC16C662 have five sources:

- External interrupt RB0/INT
- TMR0 overflow interrupt
- PORTB change interrupts (pins RB7:RB4)
- Comparator interrupt
- Parallel Slave Port interrupt (PIC16C661/662)

The interrupt control register, (INTCON), records individual core interrupt requests in flag bits. It also has various individual enable bits and the global interrupt enable bit.

The global interrupt enable bit, GIE (INTCON<7>) enables (if set) all un-masked interrupts or disables (if cleared) all interrupts. Individual interrupts can be disabled through their corresponding enable bits in INTCON register. GIE is cleared on reset.

The "return from interrupt" instruction, `RETFIE`, exits the interrupt routine as well as sets the GIE bit, which allows any pending interrupt to execute.

Those interrupts associated with the "core" have their flag and enable bits in the INTCON register. The core interrupts are: RB0/INT pin interrupt, the RB port change interrupt, and the TMR0 overflow interrupt. The INTCON register also contains the Peripheral Interrupt Enable bit, PEIE. Bit PEIE will enable/mask the peripheral interrupts (CM and PSP) from vectoring when bit PEIE is set/cleared.

Flag bits PSPIF and CMIF are contained in special function register PIR1. The corresponding interrupt enable bits (PSPIE and CMIE) are contained in special function register PIE1.

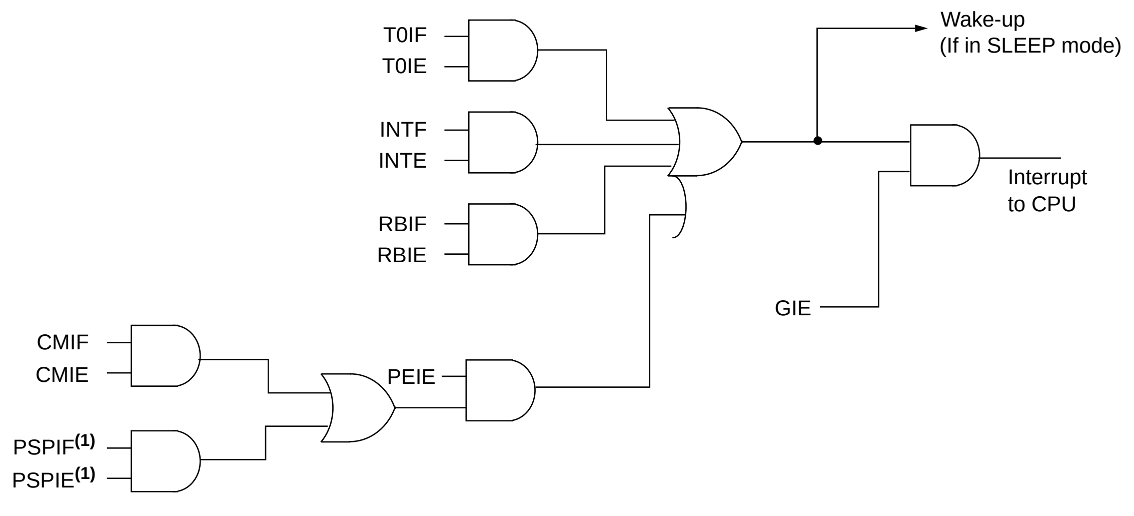
When an interrupt is responded to, the GIE is cleared to disable any further interrupt, the return address is pushed into the stack and the PC is loaded with 0004h. Once in the interrupt service routine the source(s) of the interrupt can be determined by polling the interrupt flag bits. The interrupt flag bit(s) must be cleared in software before re-enabling interrupts to avoid recursive interrupts.

For external interrupt events, such as the RB0/INT or Port RB change interrupt, the interrupt latency will be three or four instruction cycles. The exact latency depends when the interrupt event occurs (Figure 9-16). The latency is the same for one or two cycle instructions. Once in the interrupt service routine the source(s) of the interrupt can be determined by polling the interrupt flag bits. The interrupt flag bit(s) must be cleared in software before re-enabling interrupts to avoid multiple interrupt requests. Individual interrupt flag bits are set regardless of the status of their corresponding mask bit or the GIE bit.

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

Note 2: When an instruction that clears the GIE bit is executed, any interrupts that were pending for execution in the next cycle are ignored. The CPU will execute a NOP in the cycle immediately following the instruction which clears the GIE bit. The interrupts which were ignored are still pending to be serviced when the GIE bit is set again.

FIGURE 9-15: INTERRUPT LOGIC



Note 1: The Parallel Slave Port is implemented on the PIC16C661 and PIC16C662 only.

9.6 Context Saving During Interrupts

During an interrupt, only the return PC value is saved on the stack. Typically, users may wish to save key registers during an interrupt e.g. W register and STATUS register. This will have to be implemented in software.

Example 9-1 stores and restores the STATUS and W registers. The user register, W_TEMP, must be defined in both banks and must be defined at the same offset from the bank base address (i.e., W_TEMP is defined at 0x70 - 0x7F in Bank 0). The user register, STATUS_TEMP, must be defined in Bank 0.

Example 9-1:

- Stores the W register regardless of current bank
- Stores the STATUS register in Bank 0
- Executes the ISR code
- Restores the STATUS (and bank select bit register)
- Restores the W register

EXAMPLE 9-1: SAVING THE STATUS AND W REGISTERS IN RAM

```
MOVWF   W_TEMP         ; Copy W to a Temporary Register regardless of current bank
SWAPF   STATUS,W       ; Swap STATUS nibbles and place into W register
BCF     STATUS,RP0     ; Change to Bank 0 regardless of current bank
MOVWF   STATUS_TEMP    ; Save STATUS to a Temporary register in Bank 0
    :
    : (Interrupt Service Routine)
    :
SWAPF   STATUS_TEMP,W  ; Swap original STATUS register value into W (restores original bank)
MOVWF   STATUS         ; Restore STATUS register from W register
SWAPF   W_TEMP,F       ; Swap W_Temp nibbles and return value to W_Temp
SWAPF   W_TEMP,W       ; Swap W_Temp to W to restore original W value without affecting STATUS
```

9.9 Code Protection

If the code protection bit(s) have not been programmed, the on-chip program memory can be read out for verification purposes.

Note:	Microchip does not recommend code protecting windowed devices.

9.10 ID Locations

Four memory locations (2000h-2003h) are designated as ID locations where the user can store checksum or other code-identification numbers. These locations are not accessible during normal execution but are readable and writable during program/verify. Only the least significant 4 bits of the ID locations are used.

9.11 In-Circuit Serial Programming

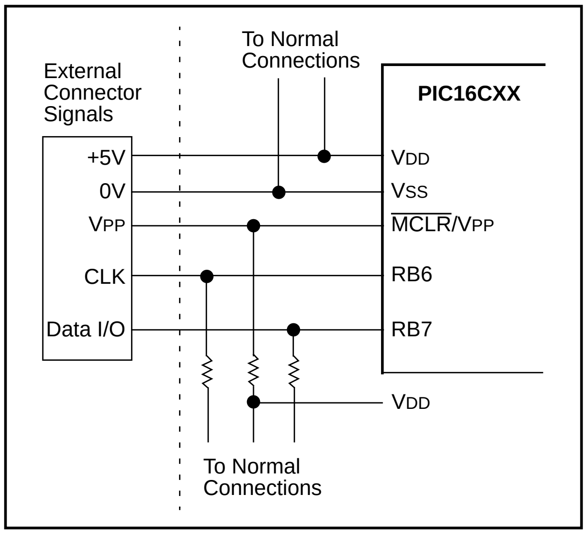
The PIC16CXX microcontrollers can be serially programmed while in the end application circuit. This is simply done with two lines for clock and data, and three other lines for power, ground, and the programming voltage. This allows customers to manufacture boards with unprogrammed devices, and then program the microcontroller just before shipping the product. This also allows the most recent firmware or a custom firmware to be programmed.

The device is placed into a program/verify mode by holding the RB6 and RB7 pins low while raising the $\overline{\text{MCLR}}$ (V_{PP}) pin from V_{IL} to V_{IHH} (see programming specification). RB6 becomes the programming clock and RB7 becomes the programming data. Both RB6 and RB7 are Schmitt Trigger inputs in this mode.

After reset, to place the device into programming/verify mode, the program counter (PC) is at location 00h. A 6-bit command is then supplied to the device. Depending on the command, 14-bits of program data are then supplied to or from the device, depending if the command was a load or a read. For complete details of serial programming, please refer to the PIC16C6X/7X Programming Specifications (Literature #DS30228).

A typical in-circuit serial programming connection is shown in Figure 9-20.

FIGURE 9-20: TYPICAL IN-CIRCUIT SERIAL PROGRAMMING CONNECTION



10.0 INSTRUCTION SET SUMMARY

Each PIC16CXX instruction is a 14-bit word divided into an OPCODE which specifies the instruction type and one or more operands which further specify the operation of the instruction. The PIC16CXX instruction set summary in Table 10-2 lists **byte-oriented**, **bit-oriented**, and **literal and control** operations. Table 10-1 shows the opcode field descriptions.

For **byte-oriented** instructions, 'f' represents a file register designator and 'd' represents a destination designator. The file register designator specifies which file register is to be used by the instruction.

The destination designator specifies where the result of the operation is to be placed. If 'd' is zero, the result is placed in the W register. If 'd' is one, the result is placed in the file register specified in the instruction.

For **bit-oriented** instructions, 'b' represents a bit field designator which selects the number of the bit affected by the operation, while 'f' represents the number of the file in which the bit is located.

For **literal and control** operations, 'k' represents an eight or eleven bit constant or literal value.

TABLE 10-1: OPCODE FIELD DESCRIPTIONS

Field	Description
`f`	Register file address (0x00 to 0x7F)
`W`	Working register (accumulator)
`b`	Bit address within an 8-bit file register
`k`	Literal field, constant data or label
`x`	Don't care location (= 0 or 1) The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
`d`	Destination select; d = 0: store result in W, d = 1: store result in file register f. Default is d = 1
`label`	Label name
`TOS`	Top of Stack
`PC`	Program Counter
`PCLATH`	Program Counter High Latch
`GIE`	Global Interrupt Enable bit
`WDT`	Watchdog Timer/Counter
$\overline{\text{TO}}$	Time-out bit
$\overline{\text{PD}}$	Power-down bit
`dest`	Destination either the W register or the specified register file location
[ ]	Options
( )	Contents
$\rightarrow$	Assigned to
< >	Register bit field
$\in$	In the set of
italics	User defined term (font is courier)

The instruction set is highly orthogonal and is grouped into three basic categories:

- **Byte-oriented** operations
- **Bit-oriented** operations
- **Literal and control** operations

All instructions are executed within one single instruction cycle, unless a conditional test is true or the program counter is changed as a result of an instruction. In this case, the execution takes two instruction cycles with the second cycle executed as a NOP. One instruction cycle consists of four oscillator periods. Thus, for an oscillator frequency of 4 MHz, the normal instruction execution time is 1 µs. If a conditional test is true or the program counter is changed as a result of an instruction, the instruction execution time is 2 µs.

Table 10-2 lists the instructions recognized by the MPASM assembler.

Figure 10-1 shows the three general formats that the instructions can have.

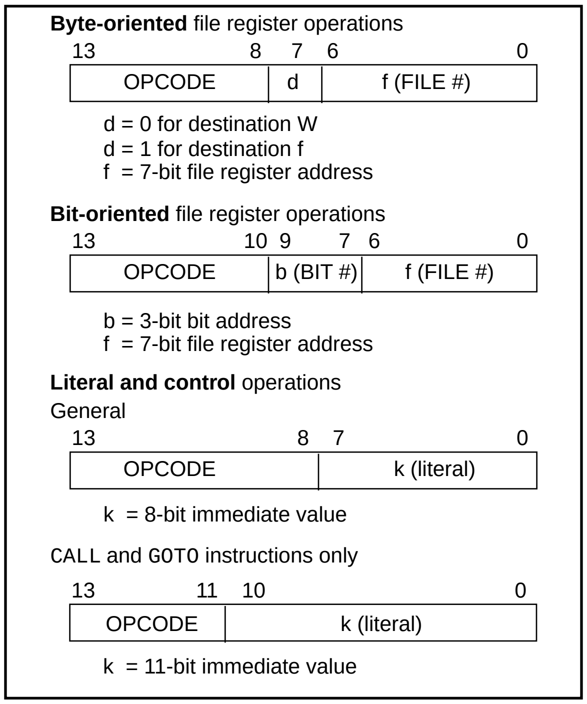
Note:	To maintain upward compatibility with future PIC16CXX products, <u>do not use</u> the `OPTION` and `TRIS` instructions.

All examples use the following format to represent a hexadecimal number:

0xhh

where h signifies a hexadecimal digit.

FIGURE 10-1: GENERAL FORMAT FOR INSTRUCTIONS


Byte-oriented file register operations
13 8 7 6 0
OPCODE
d
f (FILE #)
d = 0 for destination W
d = 1 for destination f
f = 7-bit file register address
Bit-oriented file register operations
13 10 9 7 6 0
OPCODE
b (BIT #)
f (FILE #)
b = 3-bit bit address
f = 7-bit file register address
Literal and control operations
General
13 8 7 0
OPCODE
k (literal)
k = 8-bit immediate value
CALL and GOTO instructions only
13 11 10 0
OPCODE
k (literal)
k = 11-bit immediate value

10.2 Instruction Descriptions

ADDLW	**Add Literal and W**
Syntax:	[*label*] ADDLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) + k → (W)
Status Affected:	C, DC, Z
Encoding:	`11` `111x` `kkkk` `kkkk`
Description:	The contents of the W register are added to the eight bit literal 'k' and the result is placed in the W register.
Words:	1
Cycles:	1

Example	`ADDLW 0x15`
	Before Instruction
	W = 0x10
	After Instruction
	W = 0x25

ADDWF	**Add W and f**
Syntax:	[*label*] ADDWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) + (f) → (dest)
Status Affected:	C, DC, Z
Encoding:	`00` `0111` `dfff` `ffff`
Description:	Add the contents of the W register with register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.
Words:	1
Cycles:	1

Example	`ADDWF FSR, 0`
	Before Instruction
	W = 0x17
	FSR = 0xC2
	After Instruction
	W = 0xD9
	FSR = 0xC2

ANDLW	**And Literal with W**
Syntax:	[*label*] ANDLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) .AND. (k) → (W)
Status Affected:	Z
Encoding:	`11` `1001` `kkkk` `kkkk`
Description:	The contents of W register are AND'ed with the eight bit literal 'k'. The result is placed in the W register.
Words:	1
Cycles:	1

Example	`ANDLW 0x5F`
	Before Instruction
	W = 0xA3
	After Instruction
	W = 0x03

ANDWF	**AND W with f**
Syntax:	[*label*] ANDWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) .AND. (f) → (dest)
Status Affected:	Z
Encoding:	`00` `0101` `dfff` `ffff`
Description:	AND the W register with register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.
Words:	1
Cycles:	1

Example	`ANDWF FSR, 1`
	Before Instruction
	W = 0x17
	FSR = 0xC2
	After Instruction
	W = 0x17
	FSR = 0x02

BTFSS — Bit Test f, Skip if Set

Syntax:	[*label*] BTFSS f,b
Operands:	$0 \le f \le 127$ $0 \le b < 7$
Operation:	skip if (f<b>) = 1
Status Affected:	None
Encoding:	`01` `11bb` `bfff` `ffff`
Description:	If bit 'b' in register 'f' is '1' then the next instruction is skipped. If bit 'b' is '1', then the next instruction fetched during the current instruction execution, is discarded and a NOP is executed instead, making this a 2 cycle instruction.
Words:	1
Cycles:	1(2)

Example

```
HERE    BTFSC  FLAG,1
FALSE   GOTO   PROCESS_CODE
TRUE    •
        •
        •
```

Before Instruction
PC = address `HERE`
After Instruction
if FLAG<1> = 0,
PC = address `FALSE`
if FLAG<1> = 1,
PC = address `TRUE`

CLRF — Clear f

Syntax:	[*label*] CLRF f
Operands:	$0 \le f \le 127$
Operation:	00h $\rightarrow$ (f) 1 $\rightarrow$ Z
Status Affected:	Z
Encoding:	`00` `0001` `1fff` `ffff`
Description:	The contents of register 'f' are cleared and the Z bit is set.
Words:	1
Cycles:	1

Example

```
CLRF      FLAG_REG
```

Before Instruction
FLAG_REG = 0x5A
After Instruction
FLAG_REG = 0x00
Z = 1

CALL — Call Subroutine

Syntax:	[*label*] CALL k
Operands:	$0 \le k \le 2047$
Operation:	(PC)+ 1 $\rightarrow$ TOS, k $\rightarrow$ PC<10:0>, (PCLATH<4:3>) $\rightarrow$ PC<12:11>
Status Affected:	None
Encoding:	`10` `0kkk` `kkkk` `kkkk`
Description:	Call Subroutine. First, return address (PC+1) is pushed onto the stack. The eleven bit immediate address is loaded into PC bits <10:0>. The upper bits of the PC are loaded from PCLATH. `CALL` is a two cycle instruction.
Words:	1
Cycles:	2

Example

```
HERE     CALL   THERE
```

Before Instruction
PC = Address `HERE`
After Instruction
PC = Address `THERE`
TOS = Address `HERE+1`

CLRW — Clear W

Syntax:	[*label*] CLRW
Operands:	None
Operation:	00h $\rightarrow$ (W) 1 $\rightarrow$ Z
Status Affected:	Z
Encoding:	`00` `0001` `0000` `0011`
Description:	W register is cleared. Zero bit (Z) is set.
Words:	1
Cycles:	1

Example

```
CLRW
```

Before Instruction
W = 0x5A
After Instruction
W = 0x00
Z = 1

12.5 Timing Diagrams and Specifications

FIGURE 12-2: EXTERNAL CLOCK TIMING

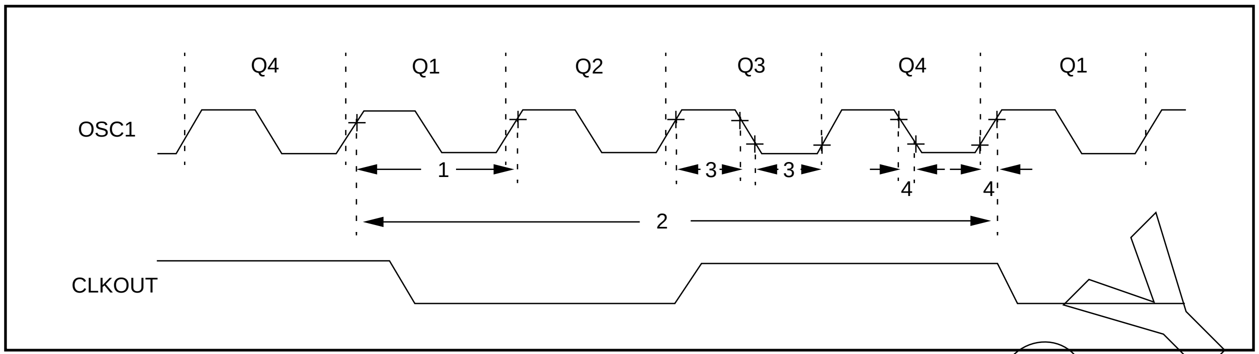


TABLE 12-4: EXTERNAL CLOCK TIMING REQUIREMENTS

Param No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
	Fosc	**External CLKIN Frequency**[(1)]	DC	—	4	MHz	XT and RC osc mode, VDD = 5.0V
			DC	—	20	MHz	HS osc mode
			DC	—	200	kHz	LP osc mode
		Oscillator Frequency [(1)]	DC	—	4	MHz	RC osc mode, VDD = 5.0V
			0.1	—	4	MHz	XT osc mode
			4	—	20	MHz	HS osc mode
			5	–	200	kHz	LP osc mode
1	Tosc	**External CLKIN Period**[(1)]	250	—	—	ns	XT and RC osc mode
			50	—	—	ns	HS osc mode
			5	—	—	µs	LP osc mode
		Oscillator Period[(1)]	250	—	—	ns	RC osc mode
			250	—	10,000	ns	XT osc mode
			50	—	250	ns	HS osc mode
			5	—	—	µs	LP osc mode
2	TCY	**Instruction Cycle Time**[(1)]	200	—	DC	ns	TCY = FOSC/4
3*	TosL, TosH	**External Clock in (OSC1) High or Low Time**	100	—	—	ns	XT osc mode
			2.5	—	—	µs	LP osc mode
			15	—	—	ns	HS osc mode
4*	TosR, TosF	**External Clock in (OSC1) Rise or Fall Time**	—	—	25	ns	XT osc mode
			—	—	50	ns	LP osc mode
			—	—	15	ns	HS osc mode

* These parameters are characterized but not tested.

† Data in “Typ” column is at 5.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: Instruction cycle period (TCY) equals four times the input oscillator time-base period. All specified values are based on characterization data for that particular oscillator type under standard operating conditions with the device executing code. Exceeding these specified limits may result in an unstable oscillator operation and/or higher than expected current consumption. All devices are tested to operate at “min.” values with an external clock applied to the OSC1 pin.
When an external clock input is used, the “Max.” cycle time limit is “DC” (no clock) for all devices.

FIGURE 12-6: TIMER0 CLOCK TIMING

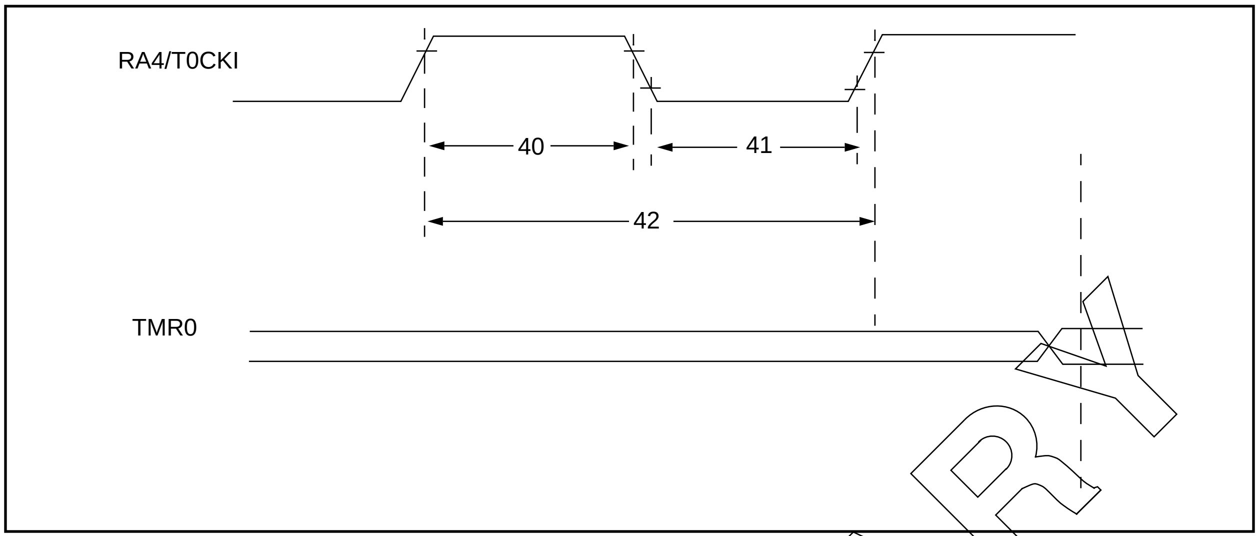


TABLE 12-7: TIMER0 CLOCK REQUIREMENTS

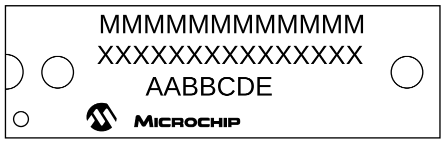
Param No.	Sym	Characteristic		Min	Typ†	Max	Units	Conditions
40*	Tt0H	T0CKI High Pulse Width	No Prescaler	0.5TCY + 20	—	—	ns	
			With Prescaler	10	—	—	ns	
41*	Tt0L	T0CKI Low Pulse Width	No Prescaler	0.5TCY + 20	—	—	ns	
			With Prescaler	10	—	—	ns	
42*	Tt0P	T0CKI Period		$\frac{T_{CY} + 40}{N}$	—	—	ns	N = prescale value (1, 2, 4, …, 256)

* These parameters are characterized but not tested.

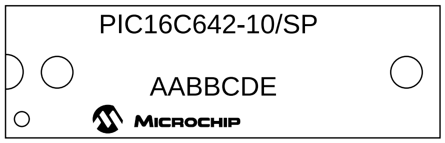
† Data in "Typ" column is at 5.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

14.1 Package Marking Information

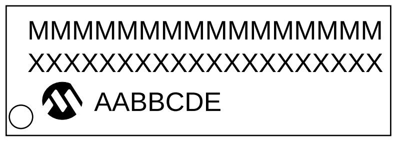
28-Lead PDIP (Skinny DIP)



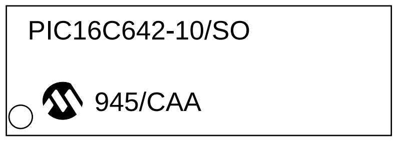
Example



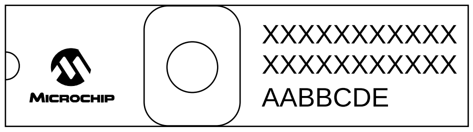
28-Lead SOIC



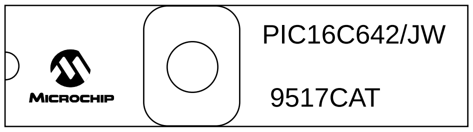
Example



28-Lead Side Brazed Skinny Windowed



Example



Legend: MM...M	Microchip part number information
XX...X	Customer specific information*
AA	Year code (last 2 digits of calendar year)
BB	Week code (week of January 1 is week '01')
C	Facility code of the plant at which wafer is manufactured C = Chandler, Arizona, U.S.A.
D	Mask revision number
E	Assembly code of the plant or country of origin in which part was assembled

Note:In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line thus limiting the number of available characters for customer specific information.

*****Standard OTP marking consists of Microchip part number, year code, week code, facility code, mask rev#, and assembly code. For OTP marking beyond this, certain price adders apply. Please check with your Microchip Sales Office. For QTP devices, any special marking adders are included in QTP price.

NOTES: