MICROCHIP

PIC16C64

40-Pin EPROM-Based 8-Bit CMOS Microcontroller

FEATURES

High-performance RISC-like CPU

- Only 35 single word instructions to learn
- All single cycle instructions (200ns) except for program branches which are two-cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200ns instruction cycle
- 14-bit wide instructions and 8-bit wide data path
- 2048 x 14 on-chip EPROM program memory
- 128 x 8 general purpose registers (SRAM)
- Interrupt capability
- 33 special function hardware registers
- Eight-level deep hardware stack
- Direct, indirect and relative addressing modes

Peripheral Features

- 33 I/O pins with individual direction control
- High current sink/source for direct LED drive
- One pin that can be configured as capture input, PWM output, or compare output
  - Capture is 16-bit, max resolution 200ns
  - Compare is 16-bit, max resolution 200ns
  - PWM resolution is 1- to 10-bit. 8-bit resolution gives 80 KHz maximum frequency and 10-bit resolution gives 20 KHz maximum frequency
- TMR1: 16-bit timer/counter (time-base for capture/compare)
- TMR2: 8-bit timer/counter with 8-bit period register (time-base for PWM), prescaler and postscaler
- TMR0: 8-bit timer/counter with 8-bit programmable prescaler
- Parallel Slave Port (PSP): 8-bit wide, with external RD, WR and CS controls (microprocessor bus interface)
- Synchronous serial port (SSP) with two modes of operation:
  - 3-wire SPI
  - I²C™/ACCESS.bus™ compatible

Special Microcontroller Features

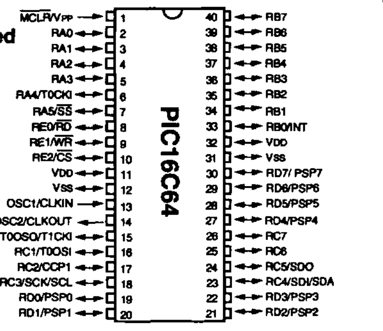
- Power-On Reset (POR)
- Power-up Timer (PWRT)
- Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Security EPROM fuses for code-protection
- Power saving SLEEP mode
- EPROM fuse selectable oscillator options: (RC oscillator, Standard crystal/resonator, High-speed crystal/resonator, Low frequency crystal)
- Serial in-system programming (via two pins)

CMOS Technology

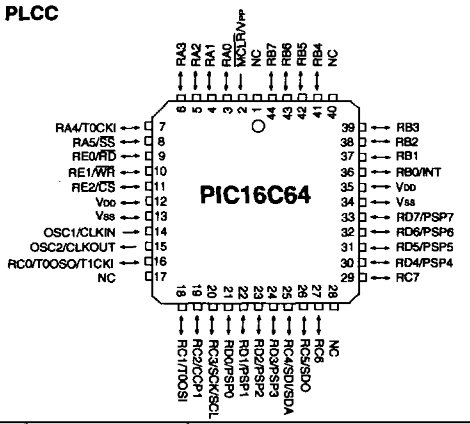
- Low-power, high-speed CMOS EPROM technology
- Fully static design
- Wide-operating voltage range (2.5V to 6.0V)
- Commercial, Industrial, and Automotive Temp. Range
- Low-power consumption
  - < 2mA @ 5V, 4 MHz
  - 15µA typical @ 3V, 32 KHz
  - < 1µA typical standby current

PACKAGE TYPES

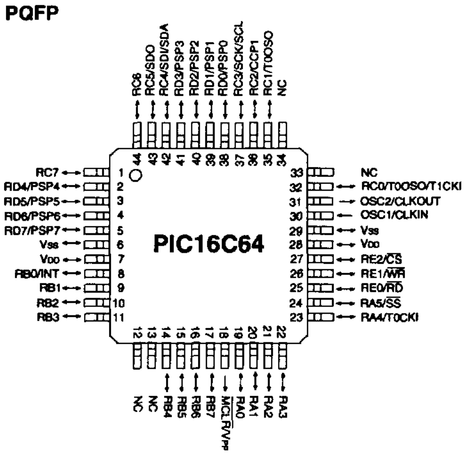
PDIP, Windowed CERDIP



PLCC



PQFP



1.0 GENERAL DESCRIPTION

The PIC16C64 is the first 4- pin member of the versatile PIC16CXX family of low-cost, high-performance, CMOS, fully static, EPROM-based 8-bit microcontrollers.

All PIC16CXX microcontrollers employ an advanced RISC-like architecture. The PIC16CXX has enhanced core features, 8-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. PIC16CXX microcontrollers typically achieve a 2:1 code compression and a 4:1 speed improvement over other 8-bit microcontrollers in its class.

The PIC16C64 has 128 bytes of RAM and 33 I/O pins. In addition, the PIC16C64 adds several peripheral features useful in many high performance applications including; three timer/counters, capture, compare, PWM features and serial ports. The synchronous serial port can be configured as either a 3-wire SPI or I^{2}C. An 8-bit Parallel slave port is also provided.

The PIC16C64 has special features which 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 reset.

A highly reliable watchdog timer with its own on-chip RC oscillator provides protection against software malfunction.

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.

Table 1-1 shows the features of the PIC16C64 as well as the other members of the PIC16CXX enhanced core family.

A simplified block diagram of the PIC16C64 is shown in Figure 3-1.

The PIC16C64 fits perfectly in applications ranging from high speed automotive and appliance motor control to low-power remote sensors, keyboards and telecom processors. The EPROM technology makes customization of application programs (transmitter codes, motor speeds, receiver frequencies, etc.) extremely fast and convenient. The small footprint packages for through hole or surface mounting 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 PIC16C64 very versatile even in areas where no microcontroller use has been considered before (e.g. timer functions, replacement of "glue" logic in larger systems, co-processor applications).

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 modifications. Code written for PIC16C5X can be easily ported to PIC16CXX family of devices (see Appendix B).

1.2 Development Support

The PIC16CXX family is supported by a full-featured macro assembler, a software simulator, an in-circuit emulator, a low-cost development programmer and a production quality programmer. A "C" compiler and fuzzy logic support tools are in development.

TABLE 1-1: PIC16CXX FAMILY OF DEVICES

		PIC16C74†	PIC16C64	PIC16C71†	PIC16C84†
Maximum Frequency of Operation		20 MHz	20 MHz	16 MHz	10 MHz
Program Memory (14-bit wide)	EPROM	4K	2K	1K	-
	EEPROM	-	-	-	1K
Data Memory (bytes)		192	128	36	36
Data EEPROM (bytes)		-	-	-	64
Timer 0 (8-bit + 8-bit prescaler)		Yes	Yes	Yes	Yes
Timer 1 (16-bit)		Yes	Yes	-	-
Timer 2 (8-bit)		Yes	Yes	-	-
Capture/Compare/PWM Module(s)		2	1	-	-
Synchronous Serial Port (SPI/I²C)		Yes	Yes	-	-
Serial Communications Interface (USART)		Yes	-	-	-
Parallel Slave Port		Yes	Yes	-	
Analog to Digital Converter (8-bit)		8 ch.	-	4 ch.	-
Power On Reset		Yes	Yes	Yes	Yes
Watchdog Timer		Yes	Yes	Yes	Yes
External Interrupts		Yes	Yes	Yes	Yes
Interrupt Sources		12	8	4	4
Program Memory Code Protect		Yes	Yes	Yes	Yes
I/O		33	33	13	13
I/O High Current Capability	Source	25mA	25mA	20mA	20mA
	Sink	25mA	25mA	25mA	25mA
Package Types		40-pin DIP, 44-pin PLCC, 44-pin PQFP	40-pin DIP, 44-pin PLCC, 44-pin PQFP	18-pin DIP 18-pin SOIC	18-pin DIP 18-pin SOIC

† For information on these devices please refer to their respective data sheets.

2.0 PIC16C64 DEVICE VARIETIES

A variety of frequency ranges and packaging options are available. Depending on application and production requirements the proper device option can be selected using the information and tables in this section. When placing orders, please use the "PIC16C64 Product Identification System" on the back page of this data sheet to specify the correct part number.

2.1 UV Erasable Devices

The UV erasable version, offered in cerdip package is optimal for prototype development and pilot programs.

The UV erasable version can be erased and reprogrammed to any of the oscillator modes etc. Microchip's PICSTART™ and PRO MATE™ programmers supports programming of the PIC16C64.

2.2 One-Time-Programmable (OTP) Devices

The availability of OTP devices is especially useful for customers expecting frequent code changes and updates.

The OTP devices, packaged in plastic packages permit the user to program them once. In addition to the program memory, the configuration fuses must be programmed.

2.3 Quick-Turnaround-Production (QTP) Devices

Microchip offers a QTP Programming Service for factory production orders. This service is made available for users who chose not to program a medium to high quantity of units and whose code patterns have stabilized. The devices are identical to the OTP devices but with all EPROM locations and fuse options already programmed by the factory. Certain code and prototype verification procedures do apply before production shipments are available. Please contact your Microchip Technology sales office for more details.

2.4 Serialized Quick-Turnaround-Production (SQTP) Devices

Microchip offers the unique programming service where a few user defined locations in each device are programmed with different serial numbers. The serial numbers may be random, pseudo-random or sequential.

Serial programming allows each device to have a unique number which can serve as an entry-code, password or ID number.

3.0 ARCHITECTURAL OVERVIEW

The high performance of the PIC16CXX family can be attributed to a number of architectural features commonly found in RISC microprocessors. To begin with, the PIC16CXX uses a Harvard architecture, in which, program and data are accessed from separate memories. This improves bandwidth over traditional Von-Neuman architecture where program and data are fetched from the same memory. Separating program and data memory further allows instructions to be sized differently than 8-bit wide data word. Instruction opcodes are 14-bit wide making it possible to have all single word instructions. A 14-bit wide program memory access bus fetches a 14-bit instruction in a single cycle. A two-stage pipeline overlaps fetch and execution of instructions (see Figure 3-1). Consequently, all instructions (35) execute in a single cycle (200ns @ 20 MHz) except for program branches.

The PIC16C64 addresses 2K x 14 program memory space, all on-chip. Program execution in microcontroller mode is internal only.

The PIC16CXX can directly or indirectly address its register files or data memory. All special function registers including the program counter are mapped in the data memory. The PIC16CXX has a fairly orthogonal (symmetrical) instruction set that makes it possible to carry out any operation on any register using any addressing mode. This symmetrical nature and lack of 'special optimal situations' make programming with the PIC16CXX simple yet efficient. In addition, the learning curve is reduced significantly.

The PIC16CXX device contains an 8-bit ALU and working register. The ALU is a general purpose arithmetic unit. It performs arithmetic and Boolean functions between data in the working register and any register file.

The ALU is 8-bit wide and capable of addition, subtraction, shift and logical operations. Unless otherwise mentioned, arithmetic operations are two's complement in nature. In two-operand instructions, typically one operand is the working register (W register). The other operand is a file register or an immediate constant. In single operand instructions, the operand is either the W register or a file register.

The W register is an 8-bit working register used for ALU operations. It is not an addressable register.

Depending on the instruction executed, the ALU may affect the values of the Carry (C), Digit Carry (DC), and Zero (Z) bits in the STATUS register. The C and DC bits operate as a $\overline{\text{borrow}}$ out bit in subtraction. See the SUBLW and SUBWF instructions for examples.

A simplified block diagram is shown in Figure 3-1, with a description of the device pins in Table 3-1.

FIGURE 3-1: PIC16C64 BLOCK DIAGRAM

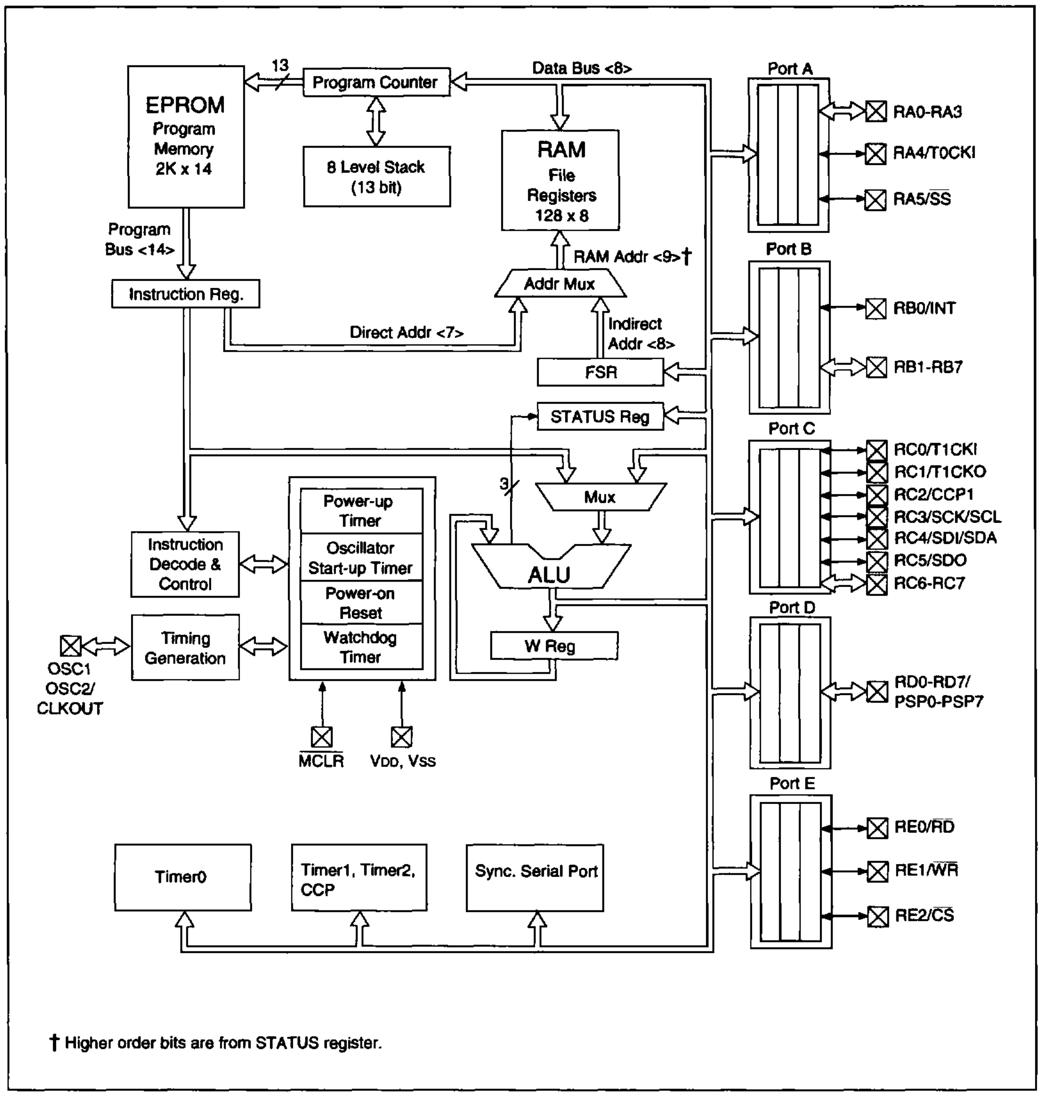


TABLE 3-1: PIC16C64 PINOUT DESCRIPTION

Pin						
Name	**DIP No.**	**PLCC No.**	**PQFP No.**	**I/O/P Type**	**Buffer Type**	**Description**
OSC1/CLKIN	13	14	30	I	CMOS	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	14	15	31	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	2	18	I/P	ST	Master clear (reset) input/programming voltage input. This pin is an active low reset to the device.
						PORTA is a bidirectional I/O port.
RA0	2	3	19	I/O	TTL	
RA1	3	4	20	I/O	TTL	
RA2	4	5	21	I/O	TTL	
RA3	5	6	22	I/O	TTL	
RA4/T0CKI	6	7	23	I/O	ST	Can also be selected to be the clock input to the TMR0 timer/counter. Output is open collector type.
RA5/$\overline{\text{SS}}$	7	8	24	I/O	TTL	Can also be the slave select for the synchronous serial port.
						PORTB is a bidirectional I/O port. Port B can be software programmed for internal weak pull-up on all inputs.
RB0/INT	33	36	8	I/O	TTL/ST†	RB0/INT can also be selected as an external interrupt pin.
RB1	34	37	9	I/O	TTL	
RB2	35	38	10	I/O	TTL	
RB3	36	39	11	I/O	TTL	
RB4	37	41	14	I/O	TTL	Interrupt on change pin.
RB5	38	42	15	I/O	TTL	Interrupt on change pin.
RB6	39	43	16	I/O	TTL/ST‡	Interrupt on change pin. Serial programming clock.
RB7	40	44	17	I/O	TTL/ST‡	Interrupt on change pin. Serial programming data.
						PORTC is a bidirectional I/O port.
RC0/T0OSO /T1CKI	15	16	32	I/O	ST	RC0/T1CKI can also be selected as a Timer1 clock/ oscillator input.
RC1/T0OSI	16	18	35	I/O	ST	RC1/T1CKO can also be selected as a Timer1 oscillator output.
RC2/CCP1	17	19	36	I/O	ST	RC2/CCP1 can also be selected as a capture1 input/ compare1 output/PWM1 output.
RC3/SCK/SCL	18	20	37	I/O	ST	RC3/SCK/SCL can also be selected as the synchronous serial clock for both SPI and I²C modes.
RC4/SDI/SDA	23	25	42	I/O	ST	RC4/SDI/SDA can also be selected as the SPI Data In (SPI mode) or data I/O (I²C mode).
RC5/SDO	24	26	43	I/O	ST	RC5/SDO can also be selected as the SPI Data Out (SPI mode).
RC6	25	27	44	I/O	ST	
RC7	26	29	1	I/O	ST	

Legend: I = input, O = output, I/O = input/output, P = power; — = Not used, TTL = TTL input, ST = Schmitt trigger input (Cont.)

† This buffer is a Schmitt triger input when configured as the external interrupt.
‡ This buffer is a Schmitt Trigger input when used in serial programming mode.

TABLE 3-1: PIC16C64 PINOUT DESCRIPTION (CONT.)

Name	Pin DIP No.	PLCC No.	PQFP No.	I/O/P Type	Buffer Type	Description
						PORTD is a bidirectional I/O port or parallel slave port for interfacing to a microprocessor bus.
RD0/PSP0	19	21	38	I/O	ST/TTL§	
RD1/PSP1	20	22	39	I/O	ST/TTL§	
RD2/PSP2	21	23	40	I/O	ST/TTL§	
RD3/PSP3	22	24	41	I/O	ST/TTL§	
RD4/PSP4	27	30	2	I/O	ST/TTL§	
RD5/PSP5	28	31	3	I/O	ST/TTL§	
RD6/PSP6	29	32	4	I/O	ST/TTL§	
RD7/PSP7	30	33	5	I/O	ST/TTL§	
RE0/$\overline{\text{RD}}$	8	9	25	I/O	ST/TTL§	Bidirectional I/O pin or read control for parallel slave port.
RE1/$\overline{\text{WR}}$	9	10	26	I/O	ST/TTL§	Bidirectional I/O pin or write control for parallel slave port.
RE2/$\overline{\text{CS}}$	10	11	27	I/O	ST/TTL§	Bidirectional I/O pin or select control for parallel slave port.
Vss	12,31	13,34	6,29	P	—	Ground reference for logic and I/O pins.
VDD	11,32	12,35	7,28	P	—	Positive supply for logic and I/O pins.
NC	—	1,17 28,40	12,13 33,34		—	These pins are not internally connected. These pins should be left unconnected.

Legend: I = input, O = output, I/O = input/output, P = power;
— = Not used, TTL = TTL input, ST = Schmitt trigger input

§ This buffer is a Schmitt trigger input when configured as general purpose I/O and a TTL input when used in the parallel slave port mode (for interfacing to a microprocessor bus).

3.1 Clocking Scheme/Instruction Cycle

The clock input (from pin OSC1) is internally divided by four to generate four non-overlapping quadrature clocks namely Q1, Q2, Q3 and Q4. Internally, the PC is incremented every Q1, the instruction is fetched from the program memory and latched into the instruction register in Q4. The instruction is decoded and executed during the following Q1 through Q4. The clocks and instruction execution flow is shown in Figure 3-2.

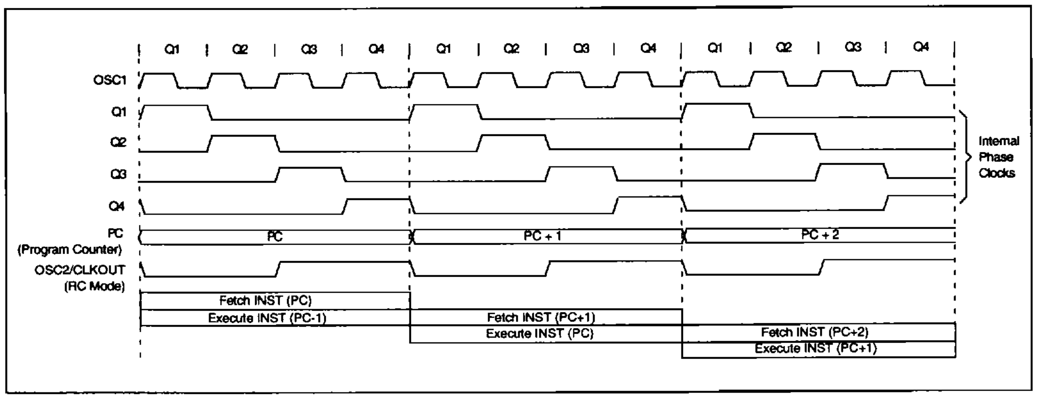
3.2 Instruction Flow/Pipelining

An "Instruction Cycle" consists of four Q cycles (Q1, Q2, Q3 and Q4). The instruction fetch and execute are pipelined such that fetch takes one instruction cycle while decode and execute takes another instruction cycle. However, due to the pipelining, each instruction effectively executes in one cycle. If an instruction causes the program counter to change (e.g. GOTO) then two cycles are required to complete the instruction (see Example 3-1).

A fetch cycle begins with the program counter (PC) incrementing in Q1.

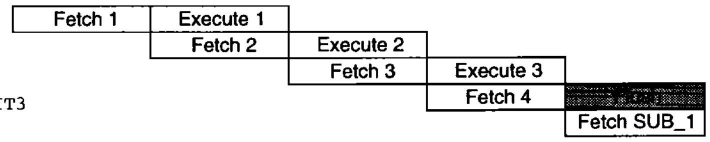
In the execution cycle, the fetched instruction is latched into the "Instruction Register (IR)" in cycle Q1. This instruction is then decoded and executed during the Q2, Q3 and Q4 cycles. Data memory is read during Q2 (operand read) and written during Q4 (destination write).

FIGURE 3-2: CLOCK/INSTRUCTION CYCLE



EXAMPLE 3-1: INSTRUCTION PIPELINE FLOW

```
1.MOVLW   55h
2.MOVWF   PORTB
3.CALL    SUB_1
4.BSF     PORTA, BIT3
```



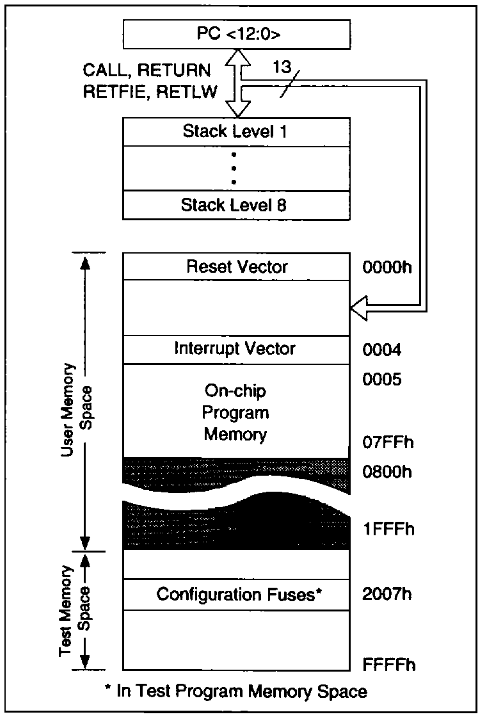
All instructions are single cycle, except for any program branches. These take two cycles since the fetched instruction is "flushed" from the pipeline while the new instruction is being fetched and then executed.

4.0 MEMORY ORGANIZATION

4.1 Program Memory Organization

The PIC16C64 has a 13-bit program counter capable of addressing an 8K x 14 program memory space. Only the first 2K x 14 (0000h - 07FFh) are physically implemented. Accessing a location above 7FFh will cause a wrap-around within the first 2K x 14 space. The reset vector is at 0000h and the interrupt vector is at 0004h (see Figure 4-1).

FIGURE 4-1: PROGRAM MEMORY MAP AND STACK



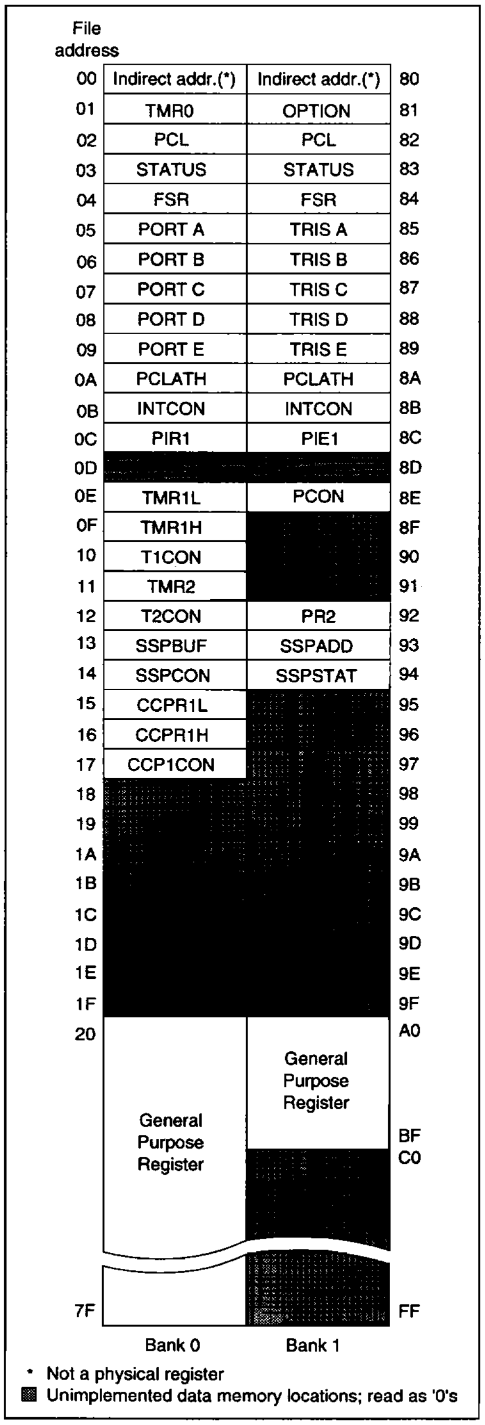
4.2 Data Memory Organization

The data memory (see Figure 4-2) is composed of the general purpose register file and the special registers. The data memory extends up to 7Fh. The first 32 locations are used to map special function registers. Locations 20h - 7Fh (Bank 0) and A0h-BFh (Bank 1) are general purpose registers implemented as static RAM. There are two register file page select bits in the STATUS register allowing selection from up to four Banks. Some special function registers are mapped in Bank 1.

4.2.1 GENERAL PURPOSE REGISTER FILE

The register file, in PIC16C64 is organized as 128 x 8. It is accessed either directly or indirectly through the file select register FSR.

FIGURE 4-2: REGISTER FILE MAP



File address	Bank 0	Bank 1	
00	Indirect addr.(*)	Indirect addr.(*)	80
01	TMR0	OPTION	81
02	PCL	PCL	82
03	STATUS	STATUS	83
04	FSR	FSR	84
05	PORT A	TRIS A	85
06	PORT B	TRIS B	86
07	PORT C	TRIS C	87
08	PORT D	TRIS D	88
09	PORT E	TRIS E	89
0A	PCLATH	PCLATH	8A
0B	INTCON	INTCON	8B
0C	PIR1	PIE1	8C
0D	▨	▨	8D
0E	TMR1L	PCON	8E
0F	TMR1H	▨	8F
10	T1CON	▨	90
11	TMR2	▨	91
12	T2CON	PR2	92
13	SSPBUF	SSPADD	93
14	SSPCON	SSPSTAT	94
15	CCPR1L	▨	95
16	CCPR1H	▨	96
17	CCP1CON	▨	97
18	▨	▨	98
19	▨	▨	99
1A	▨	▨	9A
1B	▨	▨	9B
1C	▨	▨	9C
1D	▨	▨	9D
1E	▨	▨	9E
1F	▨	▨	9F
20 – 7F	General Purpose Register	General Purpose Register	A0 – BF
		▨	C0 – FF

* Not a physical register
▨ Unimplemented data memory locations; read as '0's

4.2.2 SPECIAL FUNCTION REGISTERS:

The special function registers are registers used by the CPU and Peripheral functions for controlling the desired operation of the device (see Table 4-1). These registers are static RAM.

The special registers can be classified into two sets. The special registers associated with the "core" functions are described in this section. Those related to the operation of the peripheral features are described in the section of that peripheral feature.

TABLE 4-1: SPECIAL REGISTERS FOR THE PIC16C64

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Bank 0									
00†	INDF (Indirect Address)	Addressing this location uses contents of FSR to address data memory (not a physical register)							
01	TMR0	Timer0							
02†	PCL	Program Counter's (PC's) Least Significant Byte							
03†	STATUS	IRP	RP1	RP0	$\overline{\text{TO}}$	$\overline{\text{PD}}$	Z	DC	C
04†	FSR	Indirect data memory address pointer							
05	PORTA	PortA Data Latch where written to PortA pins when read							
06	PORTB	PortB Data Latch where written to PortB pins when read							
07	PORTC	PortC Data Latch where written to PortC pins when read							
08	PORTD	PortD Data Latch where written to PortD pins when read							
09	PORTE	PortE Data Latch where written to PortE pins when read							
0A†	PCLATH	Buffered Register for the upper 5 bits of the Program Counter (PC)							
0B†	INTCON	GIE	PEIE	RTIE	INTE	RBIE	RTIF	INTF	RBIF
0C	PIR1	PSPIF	—	—	—	SSPIF	CCP1IF	TMR2IF	TMR1IF
0D	Reserved								
0E	TMR1L	Timer1 Least Significant Byte							
0F	TMR1H	Timer1 Most Significant Byte							
10	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	T1INSYNC	TMR1CS	TMR1ON
11	TMR2	Timer2							
12	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
13	SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register							
14	SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
15	CCPR1L	Capture/Compare/Duty Cycle Register (LSB)							
16	CCPR1H	Capture/Compare/Duty Cycle Register (MSB)							
17	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0
Bank 1									
80†	INDF (Indirect Address)	Addressing this location uses contents of FSR to address data memory (not a physical register)							
81	OPTION	$\overline{\text{RBPU}}$	INTEDG	RTS	RTE	PSA	PS2	PS1	PS0
82†	PCL	Program Counter's (PC's) Least Significant Byte							
83†	STATUS	IRP	RP1	RP0	$\overline{\text{TO}}$	$\overline{\text{PD}}$	Z	DC	C
84†	FSR	Indirect data memory address pointer							
85	TRISA	PortA Data Direction Register							
86	TRISB	PortB Data Direction Register							
87	TRISC	PortC Data Direction Register							
88	TRISD	PortD Data Direction Register							
89	TRISE	IBF	OBF	IBOV	PSPMODE	—	TRISE2	TRISE1	TRISE0
8A†	PCLATH	Shadow Register for the upper 5 bits of the Program Counter (PC)							
8B†	INTCON	GIE	PEIE	RTIE	INTE	RBIE	RTIF	INTF	RBIF
8C	PIE1	PSPIE	—	—	—	SSPIE	CCP1IE	TMR2IE	TMR1IE
8D	Reserved								
8E	PCON	—	—	—	—	—	—	$\overline{\text{POR}}$	—
8F	Reserved								
90	Reserved								
91	Reserved								
92	PR2	Timer2 Period Register							
93	SSPADD	Synchronous Serial Port (I^2C mode) Address Register							
94	SSPSTAT	—	—	D/$\overline{\text{A}}$	P	S	R/$\overline{\text{W}}$	UA	BF
95	Reserved								
96	Reserved								
97	Reserved								

Legend — = Unimplemented locations, Read as '0'
† These registers can be addressed from either bank.

4.2.2.1 **STATUS** Register

The STATUS register contains the arithmetic status of the ALU, the RESET status, and the page preselect 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 logic. Furthermore, $\overline{TO}$ and $\overline{PD}$ 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` and `MOVWF` instructions are used to alter the status registers because these instructions do not affect any status bit. For other instructions, affecting any status bits, see the "Instruction Set Summary".

Note 1: The IRP and RP1 bits (STATUS<7:6>) are not used by the PIC16C64 and should be programmed as '0'. 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 $\overline{borrow}$ out bit in subtraction. See the SUBLW and SUBWF instructions for examples.

FIGURE 4-3: STATUS REGISTER

	R/W	R/W	R/W	R	R	R/W	R/W	R/W
	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
	7							bit0

Register: STATUS Address: 03h or 83h Reset value: 000??XXX $\overline{TO}$, $\overline{PD}$ are set or cleared specially	R/W: Readable & writable R: Read only U: Unimplemented, read as '0'

C: Carry/$\overline{borrow}$ bit.
For ADDWF and ADDLW instructions.
1 = A carry-out from the most significant bit of the result occurred. Note that 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.
0 = No carry-out from the most significant bit of the result.
Note: For Borrow the polarity is reversed.

DC: Digit carry/$\overline{borrow}$ bit.
For ADDWF and ADDLW instructions.
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.
Note: For Borrow the polarity is reversed.

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

$\overline{PD}$: Power down bit.
1 = After power-up or by a CLRWDT command.
0 = By execution of the SLEEP instruction.

$\overline{TO}$: Time-out bit.
1 = After power-up and by the CLRWDT and SLEEP instruction.
0 = A watchdog timer time-out has occurred.

RP<1:0>: Register page select bits for direct addressing.

00 = Bank 0 (00h - 7Fh)
01 = Bank 1 (80h - FFh)
10 = Bank 2 (100h - 17Fh)
11 = Bank 3 (180h - 1FFh)

Each page is 128 bytes.
Only the RP0 bit is used by the PIC16C64. RP1 should be programmed as '0'. Using the RP1 bit as a general purpose read/write bit is not recommended, since this may affect upward compatibility with future products.

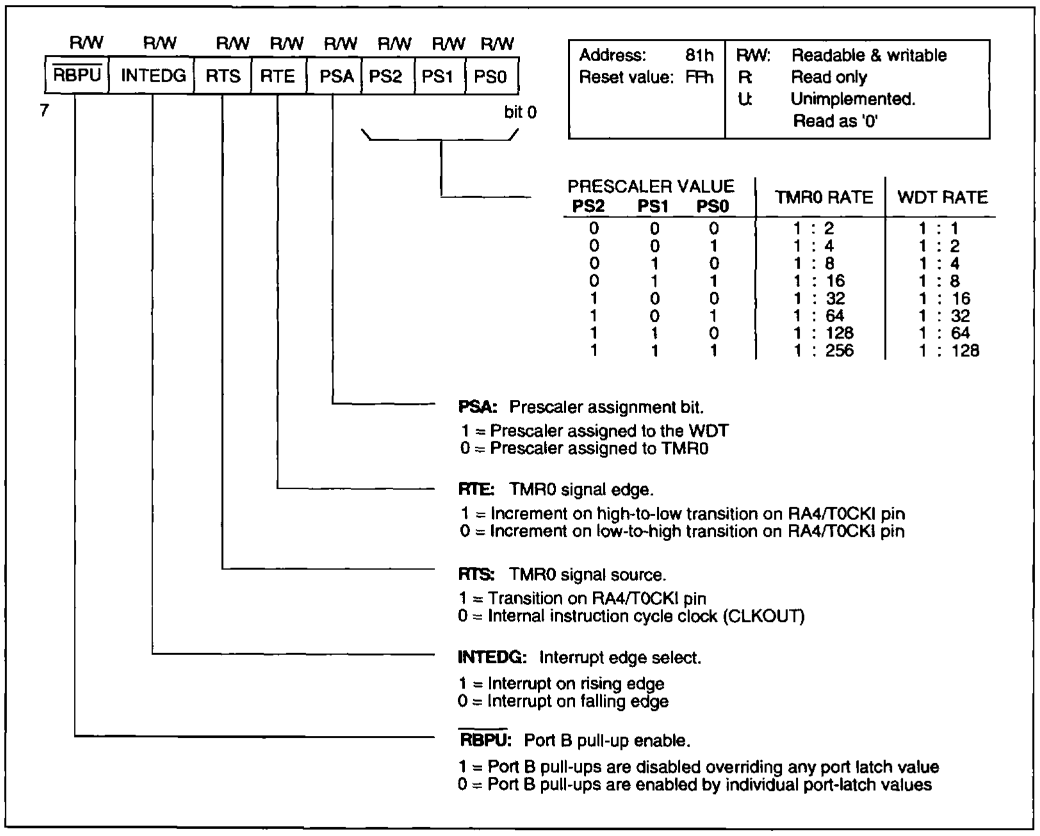
IRP: Register page select bits for indirect addressing.
0 = Bank 0,1 (00h - FFh)
1 = Bank 2,3 (100h - 1FFh)

The IRP bit is not used by the PIC16C64. IRP should be programmed as '0'. Use of the IRP bit as a general purpose read/write bit is not recommended, since this may affect upward compatibility with future products.

4.2.2.2 **OPTION** Register

The OPTION register (address 81h) is a readable and writable register which contains various control bits to configure the prescaler, the external INT interrupt, TMR0, and the weak pull-ups on PORTB.

FIGURE 4-4: OPTION REGISTER


R/W R/W R/W R/W R/W R/W R/W R/W
RBPU INTEDG RTS RTE PSA PS2 PS1 PS0
7
bit 0

Address: 81h	R/W: Readable & writable
Reset value: FFh	R: Read only
	U: Unimplemented. Read as '0'

PRESCALER VALUE			TMR0 RATE	WDT RATE
PS2	**PS1**	**PS0**		
0	0	0	1 : 2	1 : 1
0	0	1	1 : 4	1 : 2
0	1	0	1 : 8	1 : 4
0	1	1	1 : 16	1 : 8
1	0	0	1 : 32	1 : 16
1	0	1	1 : 64	1 : 32
1	1	0	1 : 128	1 : 64
1	1	1	1 : 256	1 : 128

PSA: Prescaler assignment bit.
1 = Prescaler assigned to the WDT
0 = Prescaler assigned to TMR0

RTE: TMR0 signal edge.
1 = Increment on high-to-low transition on RA4/T0CKI pin
0 = Increment on low-to-high transition on RA4/T0CKI pin

RTS: TMR0 signal source.
1 = Transition on RA4/T0CKI pin
0 = Internal instruction cycle clock (CLKOUT)

INTEDG: Interrupt edge select.
1 = Interrupt on rising edge
0 = Interrupt on falling edge

$\overline{\text{RBPU}}$: Port B pull-up enable.
1 = Port B pull-ups are disabled overriding any port latch value
0 = Port B pull-ups are enabled by individual port-latch values

4.2.2.3 **INTCON** Register

The PIC16C64 has eight sources of interrupt:

- External interrupt from RB0/INT pin
- Timer0 overflow
- Interrupt on change on RB<7:4> pins
- Timer1 overflow
- Timer2 matches period register
- A capture, a compare, or a PWM output is reset
- The synchronous serial port
- The parallel slave port read/write

The interrupt control register (INTCON) records individual interrupt requests in flag bits. It also has individual and global enable bits. The peripheral interrupt flags reside in the PIR1 register (Addr 0Ch).

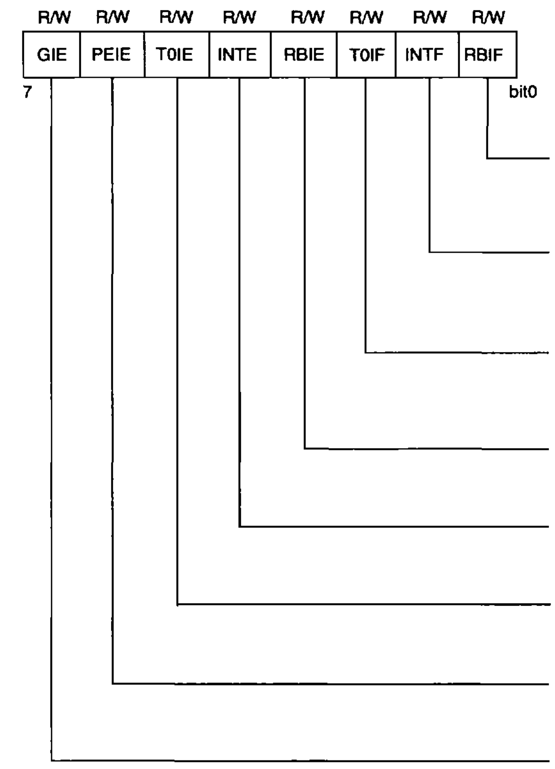
A global interrupt enable bit, GIE (INTCON<7>) enables (if set) all un-masked interrupts or disables (if cleared) all interrupts. Individual interrupts can be disabled through their corresponding mask bit in INTCON register (Figure 4-5). GIE is cleared on reset.

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. For external interrupt events, such as the INT pin or PORTB change interrupt, the interrupt latency will be 3-4 instruction cycles. The exact latency depends when the interrupt event occurs (see Figure 12-15). The latency is the same for 1 or 2 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 infinite interrupt requests. Individual interrupt flag bits are set regardless of the status of their corresponding mask bit or the GIE bit.

The "return from interrupt" instruction, RETFIE, exits interrupt routine as well as sets the GIE bit to re-enable interrupts.

Note: The T0IF, INTF, or RBIF will be set by the specified condition even if the corresponding Interrupt Enable Bit is cleared (interrupt disabled) or the GIE bit is cleared (all interrupts disabled).

FIGURE 4-5: INTCON REGISTER


R/W R/W R/W R/W R/W R/W R/W R/W
GIE PEIE T0IE INTE RBIE T0IF INTF RBIF
7
bit0

Register:	INTCON	R/W:	Readable & writable
Address:	0Bh or 8Bh	R:	Read only
Reset value:	0000 000xb	U:	Unimplemented, read as '0'

RBIF: RB port change interrupt flag.
1 = When at least one of the RB<7:4> inputs change.
Must be cleared by software.
0 = None of the RB<7:4> inputs have changed.

INTF: External interrupt flag.
1 = The external INT interrupt has occurred.
Must be cleared by software.
0 = The INT interrupt did not occur

T0IF: TMR0 overflow interrupt flag.
1 = The TMR0 has overflowed.
Must be cleared by software.
0 = TMR0 did not overflow.

RBIE: RB port change interrupt enable bit.
1 = Enables RBIF interrupt
0 = Disables RBIF interrupt

INTE: INT interrupt enable bit.
1 = Enables INTF interrupt
0 = Disables INTF interrupt

T0IE: T0IF interrupt enable bit.
1 = Enables T0IF interrupt
0 = Disables T0IF interrupt

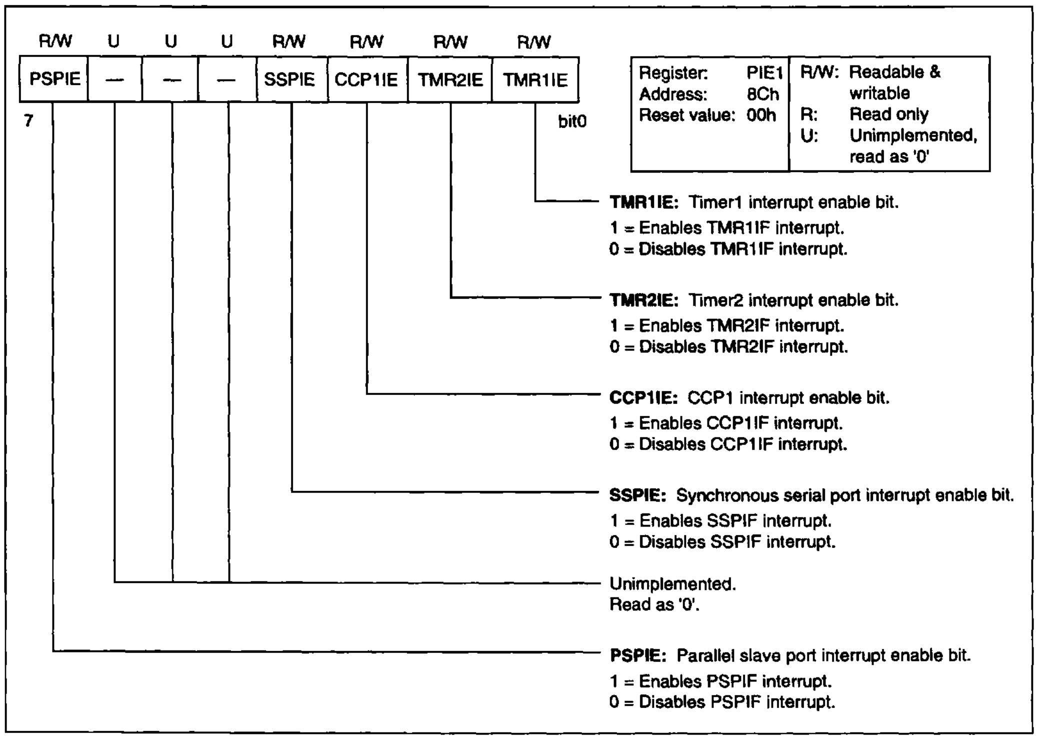
PEIE: Peripheral interrupt enable bit.
1 = Enables all un-masked peripheral interrupts
0 = Disables all peripheral interrupts

GIE: Global interrupt enable.
1 = Enables all un-masked interrupts
0 = Disables all interrupts

4.2.2.4 **PIE1** Register

This register contains the individual enable bits for the Peripheral Interrupts.

FIGURE 4-6: PIE1 REGISTER



Register:	PIE1	R/W:	Readable & writable
Address:	8Ch	R:	Read only
Reset value:	00h	U:	Unimplemented, read as '0'

TMR1IE: Timer1 interrupt enable bit.
1 = Enables TMR1IF interrupt.
0 = Disables TMR1IF interrupt.

TMR2IE: Timer2 interrupt enable bit.
1 = Enables TMR2IF interrupt.
0 = Disables TMR2IF interrupt.

CCP1IE: CCP1 interrupt enable bit.
1 = Enables CCP1IF interrupt.
0 = Disables CCP1IF interrupt.

SSPIE: Synchronous serial port interrupt enable bit.
1 = Enables SSPIF interrupt.
0 = Disables SSPIF interrupt.

Unimplemented.
Read as '0'.

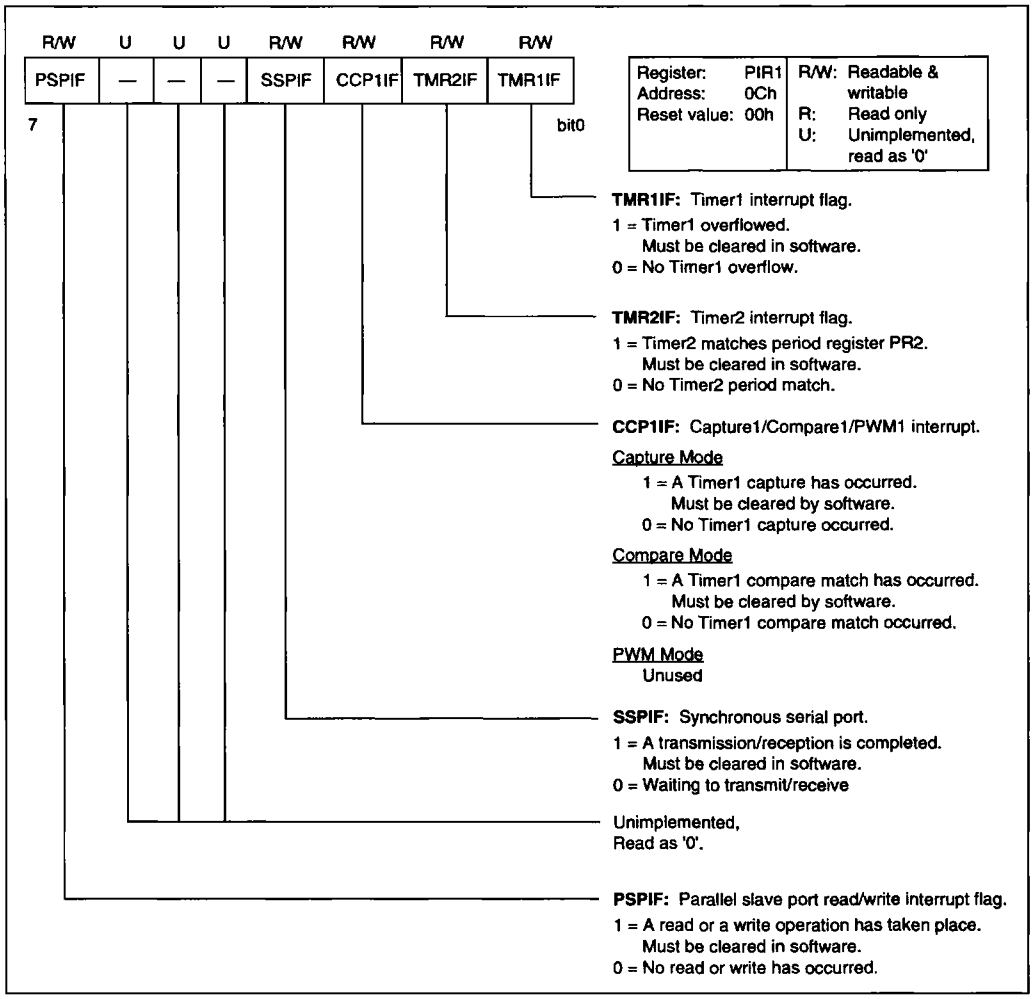
PSPIE: Parallel slave port interrupt enable bit.
1 = Enables PSPIF interrupt.
0 = Disables PSPIF interrupt.

4.2.2.5 **PIR1** Register

This register contains the individual flag bit for the Peripheral Interrupts.

Note: These bits will be set by the specified condition, even if the corresponding interrupt Enable bit is cleared (interrupt disabled) or the GIE bit is cleared (all interrupts disabled). Before enabling an interrupt, the user may wish to clear the interrupt flag, to ensure that the program does not immediately branch to the Peripheral Interrupt service routine.

FIGURE 4-7: PIR1 REGISTER



R/W	U	U	U	R/W	R/W	R/W	R/W
PSPIF	—	—	—	SSPIF	CCP1IF	TMR2IF	TMR1IF
7							bit0

Register: PIR1		R/W:	Readable & writable
Address: 0Ch		R:	Read only
Reset value: 00h		U:	Unimplemented, read as '0'

TMR1IF: Timer1 interrupt flag.
1 = Timer1 overflowed.
Must be cleared in software.
0 = No Timer1 overflow.

TMR2IF: Timer2 interrupt flag.
1 = Timer2 matches period register PR2.
Must be cleared in software.
0 = No Timer2 period match.

CCP1IF: Capture1/Compare1/PWM1 interrupt.

Capture Mode
1 = A Timer1 capture has occurred.
Must be cleared by software.
0 = No Timer1 capture occurred.

Compare Mode
1 = A Timer1 compare match has occurred.
Must be cleared by software.
0 = No Timer1 compare match occurred.

PWM Mode
Unused

SSPIF: Synchronous serial port.
1 = A transmission/reception is completed.
Must be cleared in software.
0 = Waiting to transmit/receive

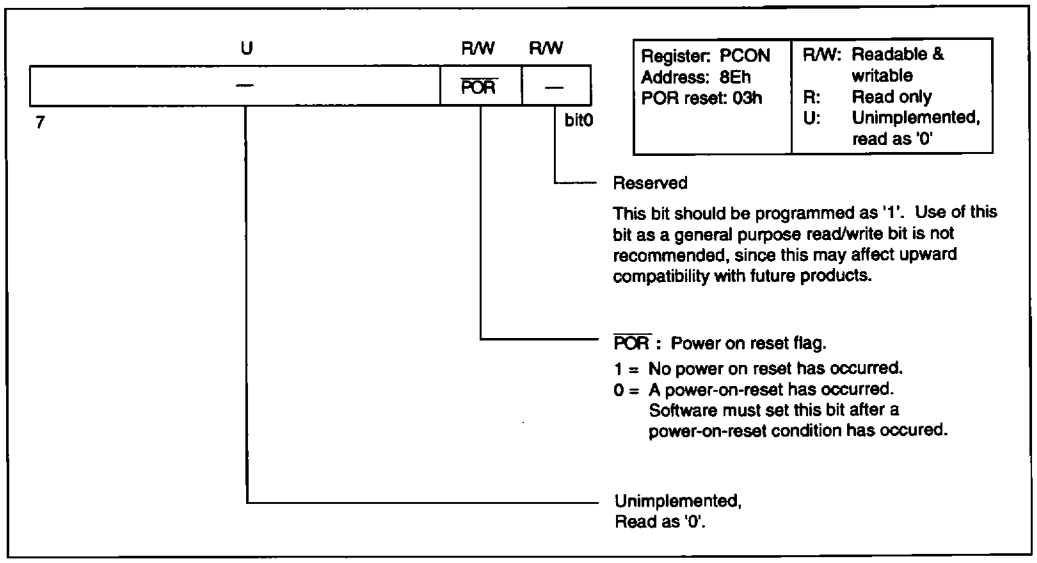
Unimplemented,
Read as '0'.

PSPIF: Parallel slave port read/write interrupt flag.
1 = A read or a write operation has taken place.
Must be cleared in software.
0 = No read or write has occurred.

4.2.2.6 **PCON** Register

The PCON register contains flag bits to allow differentiation between a Power-On Reset to an external $\overline{\text{MCLR}}$ reset, WDT reset.

FIGURE 4-8: PCON REGISTER



Register: PCON Address: 8Eh POR reset: 03h	R/W: Readable & writable R: Read only U: Unimplemented, read as '0'

Reserved

This bit should be programmed as '1'. Use of this bit as a general purpose read/write bit is not recommended, since this may affect upward compatibility with future products.

$\overline{\text{POR}}$: Power on reset flag.

1 = No power on reset has occurred.

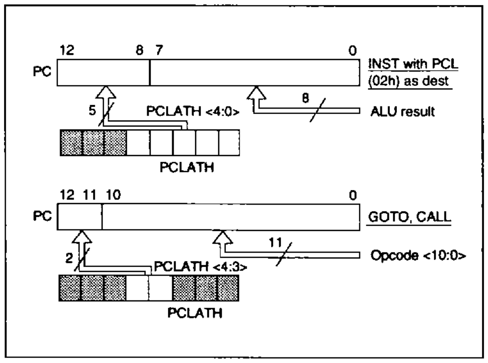
0 = A power-on-reset has occurred. Software must set this bit after a power-on-reset condition has occured.

Unimplemented, Read as '0'.

4.3 PCL and PCLATH

The program counter (PC) is 13-bit wide. The low byte, PCL is a readable and writable register (02h or 82h). The high byte of the PC, PCH is not directly readable or writable. The high byte of the PC can be written through the PCLATH register (0Ah or 8Ah). When the PC is loaded with a new value during a CALL, GOTO or a write to PCL, the high bits of PC are loaded from PCLATH, as shown in Figure 4-9.

FIGURE 4-9: LOADING OF PC IN DIFFERENT SITUATIONS



4.3.1 COMPUTED GOTO

When doing a table read using a computed GOTO method, care should be exercised if the table location crosses a PCL memory boundary (each 256 byte block). Refer to "Table Read Using the PIC16CXX" (AN556).

4.3.2 Stack

The PIC16CXX has an 8-deep x 13-bit wide hardware stack (see Figure 4-1). The stack space is not part of either program or data space and the stack pointer is not readable or writable. The PC is pushed in the stack when a CALL instruction is executed or an interrupt is acknowledged. The stack is popped in the event of a RETURN, RETLW or a RETFIE instruction execution. PCLATH is not affected by a "PUSH" or a "POP" operation.

The stack operates as a circular buffer. This means that after the stack has been "PUSHed" eight times, the ninth push overwrites the value that was stored from the first push. The tenth push overwrites the second push (and so on). This means that the user can implement "software resets" for the system.

Note: There are no STATUS bits to indicate stack overflows or stack underflow conditions.

4.4 Indirect Addressing, INDF and FSR Registers

The INDF register is not a physical register. Addressing the INDF register will cause an indirect addressing.

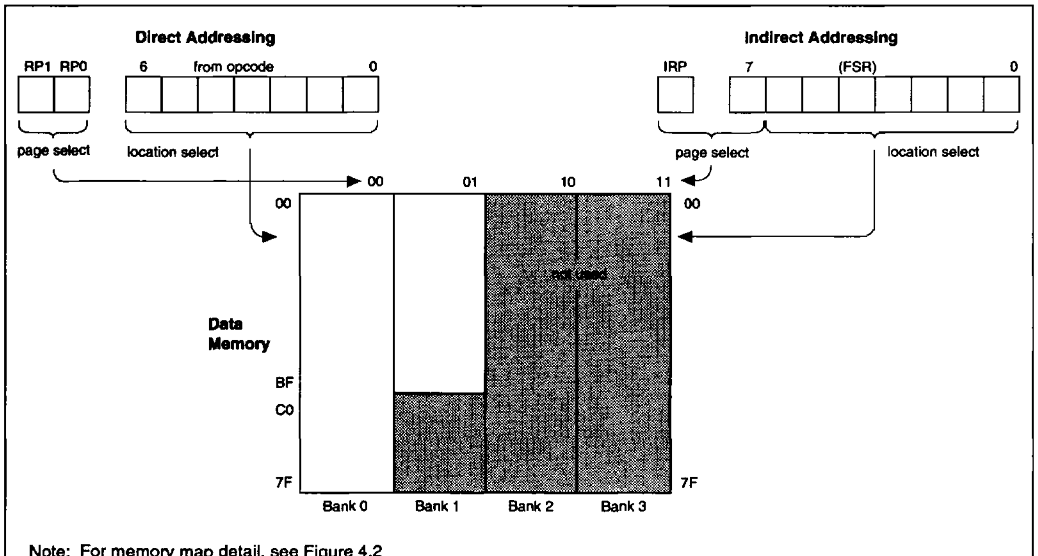
Indirect addressing is possible by using file address 00h. Any instruction using the INDF register, actually accesses data pointed to by the file select register (FSR). Reading INDF itself indirectly will produce 00h. Writing to the INDF register indirectly results in a no-operation (although status bits may be affected). An effective 9-bit address is obtained by concantenating the 8-bit FSR register and the IRP bit (STATUS<7>), as shown in Figure 4-10. However, IRP is not used in the PIC16C64.

A simple program to clear RAM location20h-2Fh using indirect addressing is shown in Example 4-1.

EXAMPLE 4-1: INDIRECT ADDRESSING

```
             movlw     0x20       ;initialize pointer
             movwf     FSR        ;to RAM
NEXT         clrf      INDF       ;clear indent by
             incrf     FSR        ;inc pointer
             btfss     FSR,4      ;all done?
             goto      NEXT       ;no clear next
CONTINUE:                         ;yes continue
```

FIGURE 4-10: DIRECT/INDIRECT ADDRESSING



Note: For memory map detail, see Figure 4.2

5.0 I/O PORTS

The PIC16C64 has five ports, PORTA through PORTE. These ports pins may be multiplexed with an alternate function for the peripheral features on the device.

5.1 PORTA and TRISA Registers

PORTA is a 6-bit wide latch. RA4 is a Schmitt trigger input and an open collector output. All other RA port pins have TTL input levels and full CMOS output drivers. All pins have data direction bits (TRIS registers) which can configure these pins as output or input.

A '1' in the TRISA register configures the corresponding port pin as input.

Port RA4 is multiplexed with TMR0 clock input.

Reading PORTA register reads the status of the pins, whereas writing to it will write to the port latch.

EXAMPLE 5-1: INITIALIZING PORTA

```
BSF     STATUS, RP0 ;Select Bank1
MOVLW   0xCF        ;Value used to initialize
                    ;data direction
MOVWF   TRISA       ;Set RA<3:0> as inputs
                    ;    RA<5:4> as outputs
                    ;TRISA<7:6> are always
                    ;read as '0'.
```

FIGURE 5-1: BLOCK DIAGRAM OF PORT PINS RA<3:0> and RA<5>

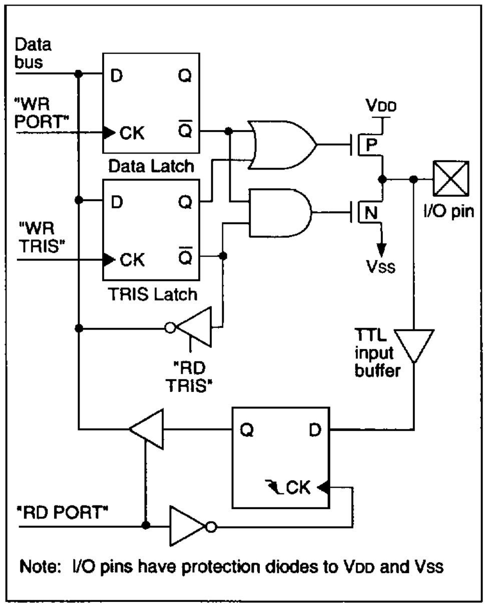


FIGURE 5-2: BLOCK DIAGRAM OF RA4 PIN

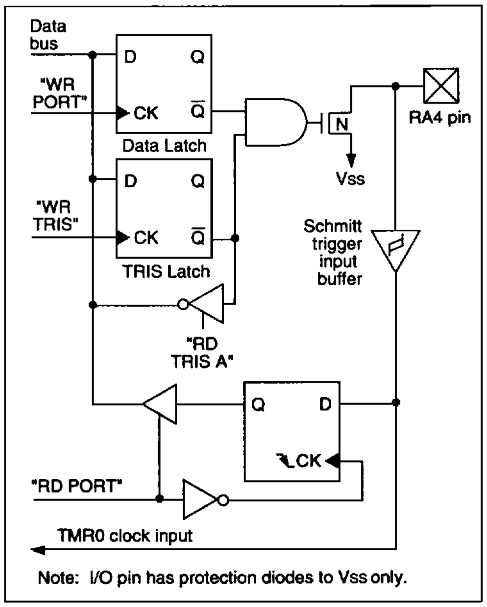


TABLE 5-1: PORTA FUNCTIONS

Name	Bit	Buffer Type	Function
RA0	bit0	TTL	Input/output
RA1	bit1	TTL	Input/output
RA2	bit2	TTL	Input/output
RA3	bit3	TTL	Input/output
RA4/T0CKI	bit4	ST	Input/output or external clock input for TMR0. Output is open collector type
RA5/$\overline{\text{SS}}$	bit5	TTL	Input/output or slave select input for synchronous serial port

Legend: TTL = TTL input, ST = Schmitt trigger input

TABLE 5-2: SUMMARY OF PORTA REGISTERS

Register Name	Function	Address	Power-on Reset Value
PORTA	PORTA pins when read PORTA latch when written	05h	`--XX XXXX`
TRISA	PORTA data direction register 0 = output, 1 = input	85h	`--11 1111`

Notes: 1: x = unknown, - = unimplemented, reads as a '0'.
2: For reset values of registers in other reset situations refer to Table 12-6.

5.2 PORTB and TRISB Registers

PORTB is an 8-bit wide bidirectional port (file register address 06h). The corresponding data direction register is TRISB (address 86h). A '1' in TRISB configures the corresponding port pin as an input. Reading PORTB register reads the status of the pins whereas writing to it will write to the port latch.

Each of the PORTB pins has a weak internal pull-up (~100 µA typical). A single control bit can turn on all the pull-ups. This is done by clearing the $\overline{\text{RBPU}}$ (OPTION<7>) bit. The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on power-on reset.

Four of PORTB's pins, RB<7:4>, have an interrupt on change feature. Only pins configured as inputs can cause this interrupt to occur (i.e. any RB7–RB4 pin configured as an output is excluded from the interrupt on change comparison). On every instruction cycle, the input pins (of RB7–RB4) are compared with the old value latched on the last read of PORTB. The "mismatch" outputs of RB7–RB4 are OR'ed together to generate the RBIF interrupt (flag latched in INTCON<0>).

This interrupt can wake the device up from SLEEP. The user, in the interrupt service routine, can clear the interrupt in one of two ways:

a) Disable the interrupt by clearing RBIE (INTCON<3>) bit.
b) Read PORTB. This will end mismatch condition. Then, clear RBIF bit.

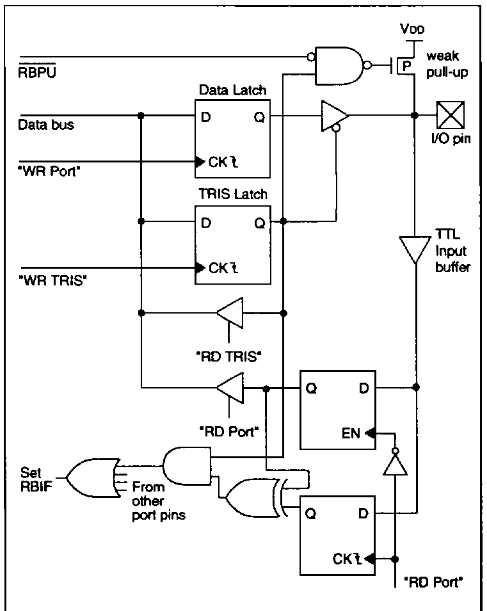
A mismatch condition will continue to set the RBIF bit. Reading PORTB will end the mismatch condition, and allow the RBIF bit to be cleared.

This interrupt on mismatch feature, together with software configurable pull-ups on these four pins allow easy interface to a key pad and make it possible for wake-up on key-depression. (See AN552 in the *Embedded Control Handbook*).

2

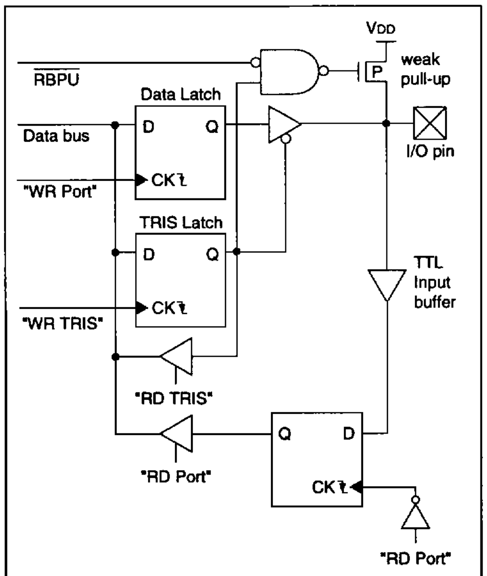
Note: If a change on the I/O pin should occur when the read operation is being executed (start of the Q2 cycle), then the RBIF interrupt flag may not be set.

FIGURE 5-3: BLOCK DIAGRAM OF PORT PINS RB<7:4>



Note: 1. Port latch = 1 and TRISB = 1 enables weak pull-up if $\overline{\text{RBPU}}$ = 0 in OPTION register.
2. I/O pins have diode protection to VDD and Vss.

FIGURE 5-4: BLOCK DIAGRAM OF PORT PINS RB<3:0>



Notes: 1. TRISB = 1 enables weak pull-up if RBPU = 0 in OPTION register.
2. For RB0/INT pin, the INT input comes through a Schmitt trigger input buffer.
3. I/O pins have diode protection to VDD and Vss.

TABLE 5-3: PORTB FUNCTIONS

Name	Bit	Buffer Type	Function
RB0/INT	bit0	TTL/ST†	Input/output pin or external interrupt input. Internal software programmable weak pull-up
RB1	bit1	TTL	Input/output pin. Internal software programmable weak pull-up
RB2	bit2	TTL	Input/output pin. Internal software programmable weak pull-up
RB3	bit3	TTL	Input/output pin. Internal software programmable weak pull-up
RB4	bit4	TTL	Input/output pin (with interrupt on change). Internal software programmable weak pull-up
RB5	bit5	TTL	Input/output pin (with interrupt on change). Internal software programmable weak pull-up
RB6	bit6	TTL/ST‡	Input/output pin (with interrupt on change). Internal software programmable weak pull-up
RB7	bit7	TTL/ST‡	Input/output pin (with interrupt on change). Internal software programmable weak pull-up

Legend: TTL = TTL input, ST = Schmitt Trigger
† This buffer is a Schmitt triger input when configured as the external interrupt.
‡ This buffer is a Schmitt Trigger input when used in serial programming mode.

TABLE 5-4: SUMMARY OF PORTB REGISTERS

Register Name	Function	Address	Power-on Reset Value
PORTB	PORTB pins when read PORTB latch when written	06h	xxxx xxxx
TRISB	PORTB data direction register 0 = output, 1 = input	86h	1111 1111
$\overline{\text{OPTION}}$	Weak pull-up on/off control ($\overline{\text{RBPU}}$ bit)	81h	1111 1111

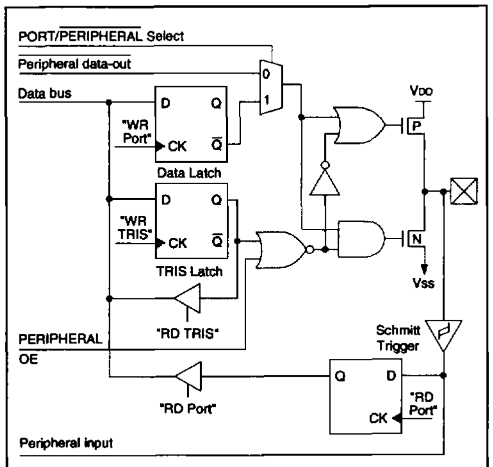
5.3 PORTC and TRISC Registers

I/O PORTC is an 8-bit bidirectional port. Each pin is individually configurable as input and output through the TRISC register. PORTC is multiplexed with several peripheral functions (see Table 5-5). PORTC pins have Schmitt trigger input buffers.

EXAMPLE 5-2: INITIALIZING PORTC

```
BSF     STATUS, RP0 ;Select Bank1
MOVLW   0xCF        ;Value used to initialize
                    ;data direction
MOVWF   TRISC       ;Set RC<3:0> as inputs
                    ;   RC<5:4> as outputs
                    ;   RC<7:6> as inputs
```

FIGURE 5-5: PORTC BLOCK DIAGRAM



Notes: 1. Port/Peripheral select signal selects between port data and peripheral output.
2. Peripheral OE (output enable) is only activated if peripheral select is active.
3. I/O pins have diode protection to VDD and Vss.

TABLE 5-5: PORTC FUNCTIONS

Name	Bit	Buffer Type	Function
RC0/T1CKI	bit0	ST	Input/output port pin or Timer1 clock input/Timer1 oscillator input
RC1/T1CKO	bit1	ST	Input/output port pin or Timer1 oscillator output
RC2/CCP1	bit2	ST	Input/output port pin or Capture1 input/Compare1 output/ PWM1 output
RC3/SCK/SCL	bit3	ST	RC3/SCK/SCL can also be selected as the synchronous serial clock for both SPI and I²C modes.
RC4/SDI/SDA	bit4	ST	RC4/SDI/SDA can also be selected as the SPI Data In (SPI mode) or data I/O (I²C mode).
RC5/SDO	bit5	ST	Input/output port pin or Synchronous serial port data output
RC6	bit6	ST	Input/output port pin
RC7	bit7	ST	Input/output port pin

Legend: ST = Schmitt Trigger Input

TABLE 5-6: SUMMARY OF PORTC REGISTERS

Register Name	Function	Address	Power-on Reset Value
PORTC	PORTC pins when read PORTC latch when written	07h	xxxx xxxx
TRISC	PORTC data direction register 0 = output, 1 = input	87h	1111 1111

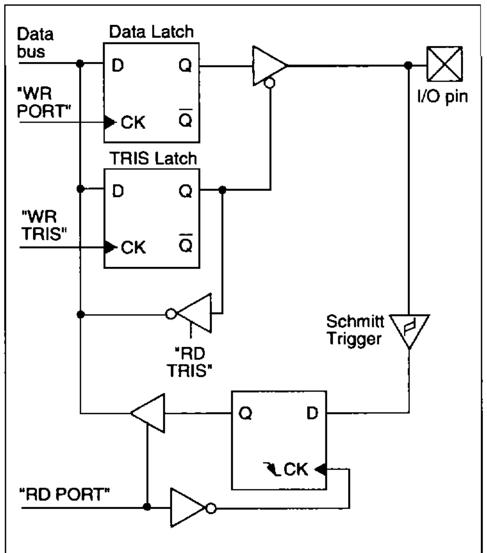
Notes: 1: x = unknown, - = unimplemented, reads as a '0'.
2: For reset values of registers in other reset situations refer to Table.

5.4 PORTD and TRISD Registers

I/O PORTD is an 8-bit port with Schmitt trigger input buffers. Each pin is individually configurable as input or output.

PORTD can be configured as an 8-bit wide microprocessor port (or parallel slave port) by setting control bit PSPMODE (TRISE<4>). In this mode, the input buffers are TTL.

FIGURE 5-6: PORTD BLOCK DIAGRAM (IN I/O PORT MODE)



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

TABLE 5-7: PORTD FUNCTIONS

Name	Bit	Buffer Type	Function
RD0/PSP0	bit0	ST/TTL	Input/output port pin or parallel slave port bit 0
RD1/PSP1	bit1	ST/TTL	Input/output port pin or parallel slave port bit 1
RD2/PSP2	bit2	ST/TTL	Input/output port pin or parallel slave port bit 2
RD3/PSP3	bit3	ST/TTL	Input/output port pin or parallel slave port bit 3
RD4/PSP4	bit4	ST/TTL	Input/output port pin or parallel slave port bit 4
RD5/PSP5	bit5	ST/TTL	Input/output port pin or parallel slave port bit 5
RD6/PSP6	bit6	ST/TTL	Input/output port pin or parallel slave port bit 6
RD7/PSP7	bit7	ST/TTL	Input/output port pin or parallel slave port bit 7

Legend: ST = Schmitt Trigger Input, TTL = TTL input

TABLE 5-8: SUMMARY OF PORTD REGISTERS

Register Name	Function	Address	Power-on Reset Value
PORTD	PORTD pins when read PORTD latch when written	08h	`xxxx xxxx`
TRISD	PORTD data direction register 0 = output, 1 = input	88h	`1111 1111`

Notes: 1: x = unknown, - = unimplemented, reads as a '0'.
2: For reset values of registers in other reset situations refer to Table.

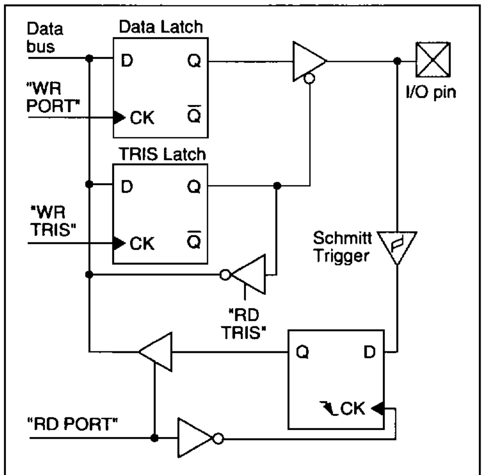
5.5 PORTE and TRISE Register

I/O PORTE has three pins RE0, RE1 and RE2 which are individually configurable as inputs or outputs. These have Schmitt trigger input buffers.

I/O PORTE becomes control inputs for the microprocessor port when the PSPMODE bit (TRISE<4>) is set. In this mode, the user must make sure that the TRISE<2:0> bits are set (pins are configured as digital inputs). In this mode the input buffers are TTL.

Figure 5-7 shows the TRISE register, which also controls the synchronous slave port operation.

FIGURE 5-7: PORTE BLOCK DIAGRAM (IN I/O PORT MODE)



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

2

FIGURE 5-8: TRISE REGISTER

R	R	R/W	R/W	U	R/W	R/W	R/W
IBF	OBF	IBOV	PSPMODE	-	TRISE2	TRISE1	TRISE0
							bit0

Register: TRISE Address: 89h POR value: 07h	R/W: Readable & writable R: Read only U: Unimplemented, read as '0'

TRISE0: Direction control bit for port pin RE0
1 = Input
0 = Output

TRISE1: Direction control bit for port pin RE1
1 = Input
0 = Output

TRISE2: Direction control bit for port pin RE2
1 = Input
0 = Output

Unimplemented.
Read as '0'

PSPMODE: Selects parallel slave port mode for ports RD and RE.
1 = Parallel slave port mode.
0 = General purpose I/O.

IBOV: Input buffer overflow in microprocessor mode.
1 = A write occurred when a previous input word has not been read.
Must be cleared in software.
0 = No overflow has occurred.

OBF: Output buffer full.
1 = The output buffer still holds a previously written word.
0 = Output buffer has been read.

IBF: Input buffer full.
1 = A word has been received and waiting to be read by the CPU.
0 = No word has been received.

TABLE 5-9: PORTE FUNCTIONS

Name	Bit	Buffer Type	Function
RE0/$\overline{RD}$	bit0	ST/TTL	Input/output port pin or Read control input in parallel slave port mode $\overline{RD}$ 1 = Not a read operation 0 = Read operation. The system reads the 16C64 PortD register (if chip selected)
RE1/$\overline{WR}$	bit1	ST/TTL	Input/output port pin or Write control input in parallel slave port mode $\overline{WR}$ 1 = Not a write operation 0 = Write operation. The system writes to the 16C64 PortD register (if chip selected)
RE2/$\overline{CS}$	bit2	ST/TTL	Input/output port pin or Chip select control input in parallel slave port mode $\overline{CS}$ 1 = Device is not selected 0 = Device is selected

Legend: ST = Schmitt Trigger Input, TTL = TTL input

TABLE 5-10: SUMMARY OF PORTE REGISTERS

Register Name	Function	Address	Power-on Reset Value
PORTE	PORTE pins when read PORTE latch when written	09h	`---- -XXX`
TRISE	PORTE data direction control bits and PORTD mode control	89h	`0000 -111`

5.6 I/O Programming Considerations

5.6.1 BIDIRECTIONAL I/O PORTS

Some instructions operate internally as read followed by write operations. The `BCF` and `BSF` instructions, for example, read the register into the CPU, execute the bit operation, and write the result back to the register. Caution must be used when these instructions are applied to a port with both inputs and outputs defined. For example, a `BSF` operation on bit 5 of PORTB will cause all eight bits of PORTB to be read into the CPU. Then the `BSF` operation takes place on bit 5 and PORTB is written to the output latches. If another bit of PORTB is used as a bidirectional I/O pin (for example, bit 0) and it is defined as an input at this time, the input signal present on the pin itself would be read into the CPU and re-written to the data latch of this particular pin, overwriting the previous content. As long as the pin stays in the input mode, no problem occurs. However, if bit 0 is switched into output mode later on, the content of the data latch may now be unknown.

A pin actively outputting a Low or High should not be driven from external devices at the same time in order to change the level on this pin ("wired-or", "wired-and"). The resulting high output currents may damage the chip.

Reading the PORT register, reads the values of the PORT pins. Writing to the PORT register writes the value to the PORT latch. When using read modify write instructions (ex. `BCF`, `BSF`, etc.) on a PORT, the value of the PORT pins is read, the desired operation is done to this value, and this value is then written to the PORT latch.

Example 5-3 shows the effect of two sequential read modify write instructions (ex. `BCF`, `BSF`, etc.) on an I/O PORT.

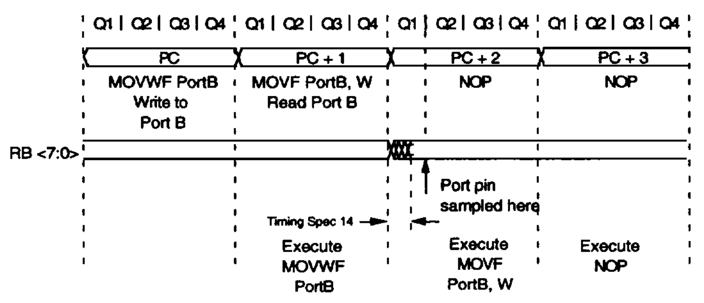
EXAMPLE 5-3: READ MODIFY WRITE INSTRUCTIONS ON AN I/O PORT

```
; Initial PORT settings:  PORTB<7:4> Inputs
;                         PORTB<3:0> Outputs
; PORTB<7:6> have external pull-up and are not
; connected to other circuitry.
;
;                             PORT latch   PORT pins
;                             ----------   ----------
      BCF     PORTB, 7      ; 01pp pppp    11pp pppp
      BCF     PORTB, 6      ; 10pp pppp    11pp pppp
      BSF     STATUS, RP0   ;
      BCF     TRISB, 7      ; 10pp pppp    11pp pppp
      BCF     TRISB, 6      ; 10pp pppp    10pp pppp
;
; Note that the user may have expected the pin
; values to be 00pp pppp. The 2nd BCF caused RB7
; to be latched as the pin value (High).
```

5.6.2 SUCCESSIVE OPERATIONS ON I/O PORTS

The actual write to an I/O port happens at the end of an instruction cycle, whereas for reading, the data must be valid at the beginning of the instruction cycle (see Figure 5-9). Therefore, care must be exercised if a write followed by a read operation is carried out on the same I/O port. The sequence of instructions should be such to allow the pin voltage to stabilize (load dependent) before the next instruction which causes that file to be read into the CPU is executed. Otherwise, the previous state of that pin may be read into the CPU rather than the new state. When in doubt, it is better to separate these instructions with a NOP or an other instruction not accessing this I/O port.

FIGURE 5-9: SUCCESSIVE I/O OPERATION


Q1 | Q2 | Q3 | Q4 | Q1 | Q2 | Q3 | Q4 | Q1 | Q2 | Q3 | Q4 | Q1 | Q2 | Q3 | Q4
PC
PC + 1
PC + 2
PC + 3
MOVWF PortB Write to Port B
MOVF PortB, W Read Port B
NOP
NOP
RB <7:0>
Port pin sampled here
Timing Spec 14
Execute MOVWF PortB
Execute MOVF PortB, W
Execute NOP

Note:

This example shows write to PortB followed by a read from PortB.
Note that :
data setup time = (0.25 TCY -TPD)
where TCY = instruction cycle.

Therefore, at higher clock frequencies, a write followed by a read may be problematic.

5.7 Parallel Slave Port

PORTD operates as an 8-bit wide parallel slave port, or 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 (RE0/$\overline{\text{RD}}$) and $\overline{\text{WR}}$ control input (RE1/$\overline{\text{WR}}$).

It can directly interface to an 8-bit microprocessor data bus. The microprocessor can read or write the PORTD latch as an 8-bit latch. Setting PSPMODE also enables the port pin RE0 to be the $\overline{\text{RD}}$ input, RE1 to be the $\overline{\text{WR}}$ input and RE2 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).

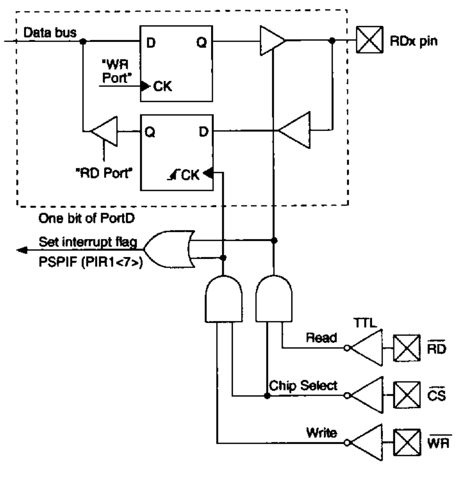
There are actually two 8-bit latches, one for data-out (from the PIC16CXX) and one for data input. The user writes 8-bit data to PORTD data latch and reads data from the port pin latch (note that they have the same address). In this mode the TRISD register is ignored, since the microprocessor is controlling the direction of data flow.

Status flag IBF, Input Buffer Full (TRISE<7>), is set if a received word is waiting to be read by the CPU. The IBF is cleared when PORTD is read by the CPU. IBF is a read only status bit. Status flag OBF, Output Buffer Full (TRISE<6>), is set if a word written to PORTD latch is waiting to be read by the external bus. The OBF flag is cleared when PORTD is read by the external device. Status flag IBOV, Input Buffer Overflow (TRISE<5>), is set if a second word is written to the microprocessor port when the previous word has not been read by the CPU. It is a read/write bit and must be cleared by the CPU.

When not in PSPMODE, IBF and OBF bits are held as cleared. However, if the IBOV flag was previously set, it must be cleared in the software.

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

FIGURE 5-10: PORTD AND PORTE AS A PARALLEL SLAVE PORT



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

TABLE 5-11: SUMMARY OF PARALLEL SLAVE PORT REGISTERS

Register Name	Function	Address	Power-on Reset Value
PORTD	Parallel slave port Read/Write Data	08h	xxxx xxxx
TRISD	PORTD data direction register	88h	1111 1111
PORTE	Parallel slave port Read/Write/Chip Select signals	09h	---- -xxx
TRISE	Control bits for PORTD peripheral	89h	0000 -111
PIR1	Interrupt register (PSPIF bit)	0Ch	0000 0000
PIE1	Interrupt Enable register (PSPIE bit)	8ch	0000 0000

6.0 OVERVIEW OF TIMER MODULES

The PIC16C64 has three timer modules. Each module can generate an interrupt to indicate that an event has occurred (i.e. timer overflow). Each of these modules is explained in full detail in the following sections. The timer modules are:

- Timer0 (TMR0) module (see Section 7)
- Timer1 (TMR1) module (see Section 8)
- Timer2 (TMR2) module (see Section 9)

For enhanced time-based functionality, the following module can be used with either of the TMR1 or TMR2 modules. This module is the:

- Capture/Compare/PWM (CCP1) module (see Section 10)

6.1 Timer0 (TMR0) Overview

The TMR0 module is identical to the RTCC module of other PIC16CXX enhanced core products (and very similar to the PIC16C5X products). The TMR0 module is a simple 8-bit overflow counter. The clock source can be either the internal system clock (OSC/4) or an external clock. When the clock source is an external clock, the TMR0 module can be selected to increment on either the rising or falling edge.

The TMR0 module also has a programmable prescaler option. This prescaler can be assigned to either the RTCC module or the Watchdog Timer. The PSA bit (OPTION<3>) assigns the prescaler, and the PS2 -PS0 (OPTION<2:0>) determines the prescaler value. The RTCC can increment at the following rates: 1:1 (when prescaler assigned to Watchdog Timer), 1:2, 1:4, 1:8, 1:16, 1:32, 1:64, 1:128, 1:256.

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

When the clock source is an external clock, the TMR0 module can be selected to increment on either the rising or falling edge.

6.2 Timer1 (TMR1) Overview

Timer1 (TMR1) is a 16-bit timer/counter. The clock source can be either the internal system clock (OSC/4), an external clock, or an external crystal. TMR1 can operate as either a timer or a counter. When operating as a counter (external clock source), the counter can either operate synchronized to the device or asynchonously to the device. Asynchonous operation allows TMR1 to operate during sleep, which is useful for applications that require a real time clock as well as the power savings of sleep mode.

TMR1 also has a prescaler option which allows the TMR1 to increment at the following rates: 1:1, 1:2, 1:4, 1:8. TMR1 can be used in conjunction with the Capture / Compare / PWM (CCP1) module. When used with the CCP1 module, TMR1 is the timebase for 16-bit capture or the 16-bit compare. When using the TMR1 module with the CCP1 module, TMR1 must be synchronized to the device (timer or synchronized counter mode).

6.3 Timer2 (TMR2) Overview

Timer2 (TMR2) is an 8-bit timer. TMR2 has both a programmable prescaler and postscaler, as well as an 8-bit period register (PR2). TMR2 can be used with the CCP1 module as well as the baud rate generator for the Synchronous Serial Port (SSP). The prescaler option which allows the TMR2 to increment at the following rates: 1:1, 1:4, 1:16.

The postscaler allows TMR2 to match the period register (PR2) a programmable number of times before generating an interrupt. The postscaler can be programmed from 1:1 to 1:16 (inclusive).

6.4 CCP1 Overview

The CCP1 module can operate in one of these three modes: 16-bit capture, 16-bit compare, or 10-bit Pulse Width Modulation (PWM)

Capture mode, captures the 16-bit value of TMR1 into the CCPR1H:CCPR1L register pair. The capture event can be programmed for either the falling edge, rising edge, fourth rising edge, or the sixteenth rising edge of the CCP1 pin.

Compare mode, compares the TMR1H:TMR1L register pair to the CCPR1H:CCPR1L register pair. When a match occurs an interrupt can be generated, and the output pin CCP1 can be forced to given state (High or Low) or TMR1 can be reset. This depends on the control bits CCP1M3 - CCP1M0.

PWM mode, compares TMR2 to a 10-bit duty cycle register as well as to an 8-bit period register (PR2). When the TMR2 = PR2, TMR2 is reset to 00h, an interrupt can be generated, and the CCP1 pin (if an output) will be forced high. When the TMR2 = Duty Cycle register, the CCP1 pin will be forced low.

7.0 TIMER0 (TMR0) MODULE

The TMR0 module timer/counter has the following features:

- 8-bit timer/counter
- Readable and writable (file address 01h)
- 8-bit software programmable prescaler
- Internal or external clock select
- Interrupt on overflow from FFh to 00h
- Edge select for external clock

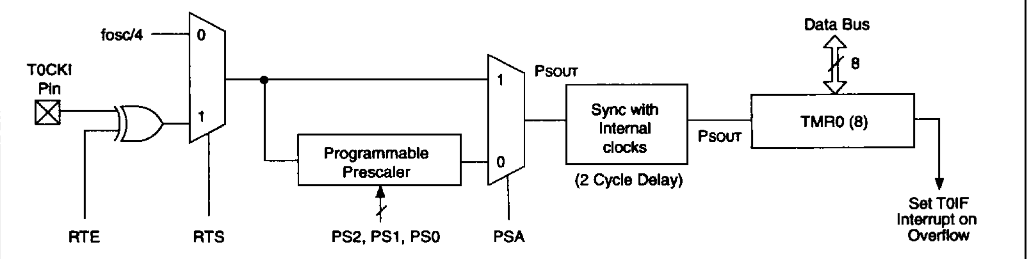
Figure 7-1 is a simplified block diagram of the TMR0 module.

Timer mode is selected by clearing the RTS bit (OPTION<5>). In timer mode, the TMR0 module will increment every instruction cycle (without prescaler). If TMR0 is written, increment is inhibited for the following two cycles (see Figures 7-2 and 7-3). The user can work around this by writing an adjusted value to the TMR0 module.

Counter mode is selected by setting the RTS bit (OPTION<5>). In this mode TMR0 will increment either on every rising or falling edge of pin RA4/T0CKI. The incrementing edge is determined by control bit RTE (OPTION<4>). Clearing the RTE bit selects the rising edge. Restrictions on the external clock input is discussed in detail in Section 7.2.

The prescaler is shared between the TMR0 module and the Watchdog Timer. The prescaler assignment is controlled in software by control bit, PSA (OPTION<3>). Clearing the PSA bit will assign the prescaler to TMR0. The prescaler is not readable or writable. When the prescaler is assigned to the TMR0 module, prescale values of 1:2, 1:4, ..., 1:256 are selectable. Section 7.3 details the operation of the prescaler.

FIGURE 7-1: TIMER0 (TMR0) BLOCK DIAGRAM



Notes: 1. Bits, RTE, RTS, PS2, PS1, PS0 AND PSA are located in OPTION register.
2. The prescaler is shared with Watchdog Timer (refer to Figure 7–6 for detailed diagram).

FIGURE 7-2: TIMER0 (TMR0) TIMING: INTERNAL CLOCK/NO PRESCALE

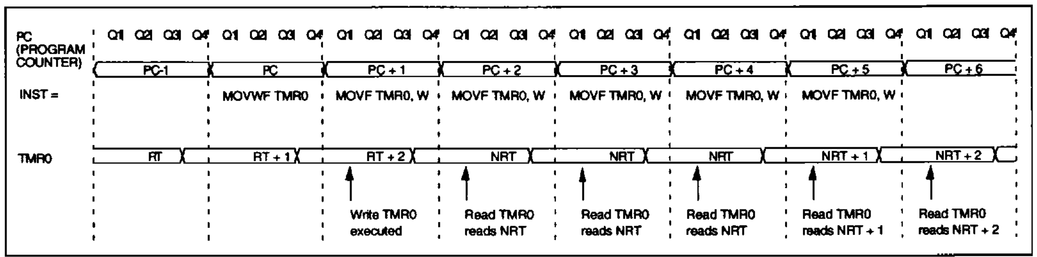
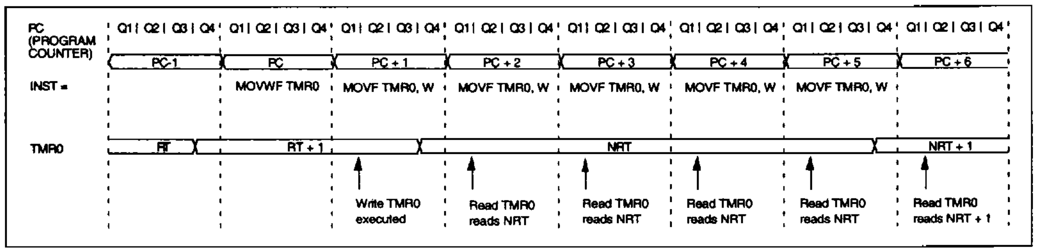


FIGURE 7-3: TIMER0 (TMR0) TIMING: INTERNAL CLOCK/PRESCALE 1:2



7.1 TIMER0 (TMR0) Interrupt

TMR0 interrupt is generated when the TMR0 module timer/counter overflows from FFh to 00h. This overflow sets the T0IF bit. The interrupt can be masked by clearing the T0IE bit (INTCON<5>). The T0IF bit (INTCON<2>) must be cleared in software by the TMR0 module interrupt service routine before re-enabling this interrupt. The TMR0 module interrupt can not wake the processor from SLEEP since the timer is shut off during SLEEP. See Figure 7-4 for TMR0 interrupt timing.

7.2 Using TMR0 with External Clock

When external clock input is used for TMR0, it is synchronized with the internal phase clocks. Therefore, the external clock input must meet certain requirements.

Also there is some delay from the occurance of the external clock edge to the actual incrementing of TMR0. Referring to Figure 7-5, the synchronization is done after the prescaler. The output of the prescaler is sampled twice in every instruction cycle to detect rising or falling edges. Therefore, it is necessary for PSOUT to be high for at least 2 tosc and low for at least 2 tosc where:

tosc = oscillator time period.

When no prescaler is used, PSOUT (Prescaler output, see Figure 7-1) is the same as TMR0 clock input and therefore the requirements are:

TRTH = TMR0 high time $\geq$ 2tosc + ΔT (See parameter #40)

TRTL = TMR0 low time $\geq$ 2tosc + ΔT (See parameter #41)

When prescaler is used, the TMR0 module input is divided by the asynchronous ripple counter-type prescaler and so the prescaler output is symmetrical.

Then:

$$\text{PSOUT high time} = \text{PSOUT low time} = \frac{N \bullet T_{RT}}{2}$$

where

TRT = TMR0 input period

N = prescale value (2, 4, ..., 256).

The requirement is, therefore:

$$\frac{N \bullet T_{RT}}{2} \geq 2\,tosc + \Delta T, \text{ or } T_{RT} \geq \frac{4\,tosc + 2\,\Delta T}{N}$$

where

ΔT = small RC delay (see Timing Specifications).

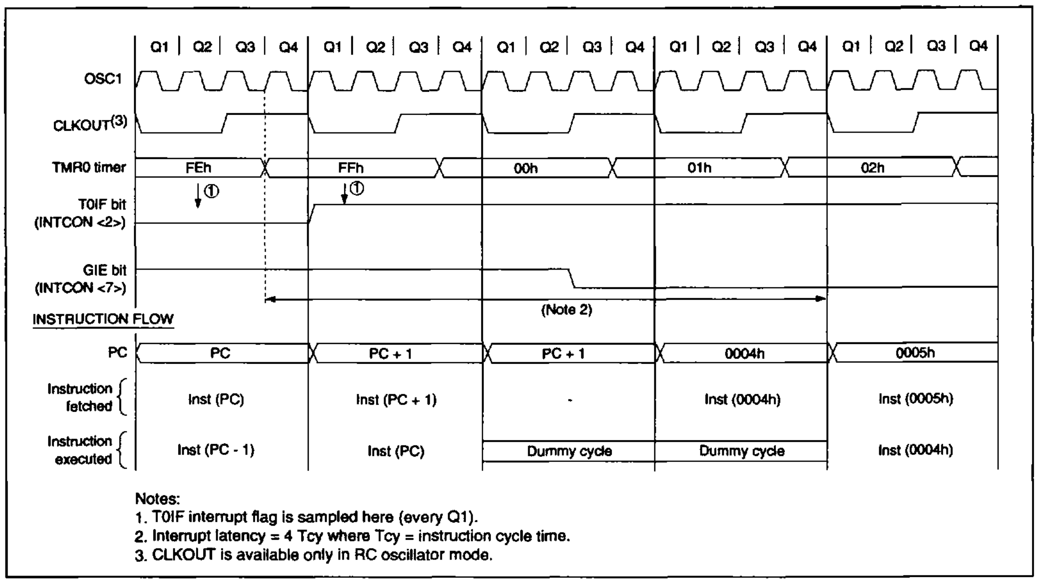
The user will notice that no requirement on TMR0 high time or low time is specified. However, if the high time or low time on TMR0 is too small then the pulse may not be detected, hence a minimum high or low time of 10 ns is required. In summary, the TMR0 module input requirements are:

TRT = TMR0 period $\geq$ (4 tosc + 2 ΔT)/N

TRTH = TMR0 high time $\geq$ ΔT

TRTL = TMR0 low time $\geq$ ΔT

FIGURE 7-4: TIMER0 (TMR0) INTERRUPT TIMING



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 the TMR0 module is actually incremented. Referring to Figure 7-5, the reader can see that this delay is between 3 tosc and 7 tosc. Thus, for example, measuring the interval between two edges (e.g. period) will be accurate within ±4 tosc (±200ns @ 20 MHz).

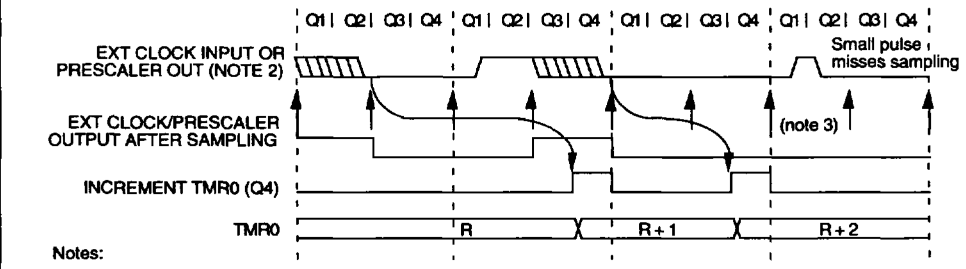
7.3 Prescaler

An 8-bit counter is available as a prescaler for the TMR0 module, or as a post-scaler for the Watchdog Timer, respectively (see Figure 7-6). For simplicity, this counter is being referred to as "prescaler" throughout this data sheet. Note that there is only one prescaler available which is mutually exclusively shared between the TMR0 module and the Watchdog Timer. Thus, a prescaler assignment for the TMR0 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 pre-scale ratio.

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

FIGURE 7-5: TIMER0 TIMING WITH EXTERNAL CLOCK

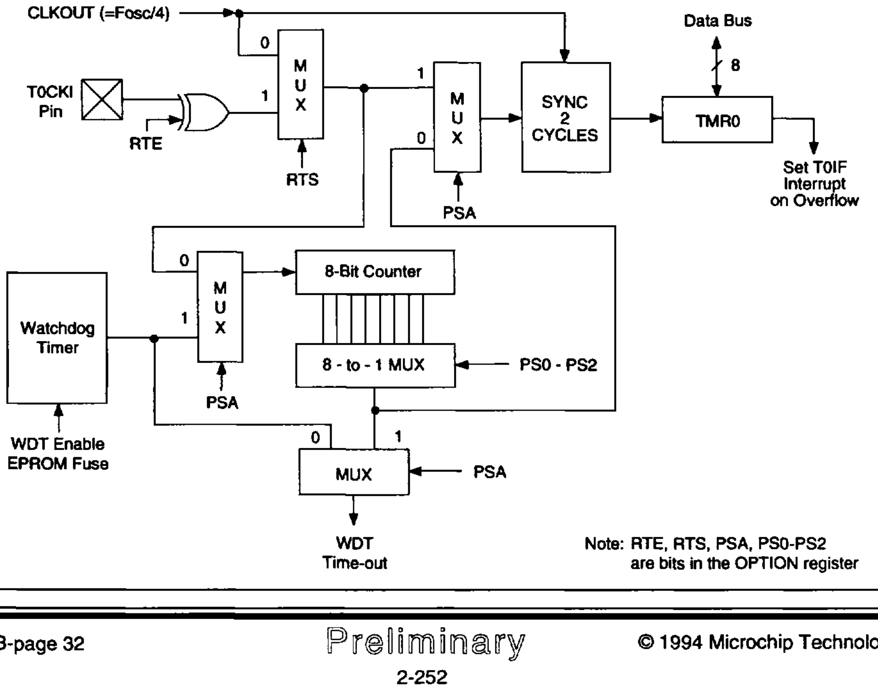


Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4
EXT CLOCK INPUT OR PRESCALER OUT (NOTE 2)
Small pulse misses sampling
EXT CLOCK/PRESCALER OUTPUT AFTER SAMPLING
(note 3)
INCREMENT TMR0 (Q4)
TMR0
R
R+1
R+2

Notes:

1. Delay from clock input change to TMR0 increment is 3 tosc to 7 tosc. (Duration of Q = tosc). Therefore, the error in measuring the interval between two edges on TMR0 input = ± 4 tosc max.
2. External clock if no prescaler selected, Prescaler output otherwise.
3. The arrows indicate the points in time where sampling occurs.

FIGURE 7-6: BLOCK DIAGRAM OF THE TMR0/WDT PRESCALER



Note: RTE, RTS, PSA, PS0-PS2 are bits in the OPTION register

7.3.1 SWITCHING PRESCALER ASSIGNMENT

The prescaler assignment is fully under software control, i.e., it can be changed "on the fly" during program execution. To avoid an unintended device RESET, the following instruction sequence (shown in Example 7-1) must be executed when changing the prescaler assignment from TMR0 to WDT. Depending on the selected prescaler value (lines 2 and 3) determines if lines 9 and 10 are required:

EXAMPLE 7-1: CHANGING PRESCALER (TMR0→WDT)

```
1. BCF     STATUS, RP0   ;Bank 0
2. CLRF    TMR0          ;Clear TMR0
3. BSF     STATUS, RP0   ;Bank 1
4. CLRWDT                ;Clears WDT and
                         ;  prescale
5. MOVLW   B'xxxx1xxx'   ;Select new prescale
6. MOVWF   OPTION        ;  value
7. BCF     STATUS, RP0   ;Bank 0
```

Steps 2 and 3 are only required if an external TMR0 source is used. Steps 9 and 10 are necessary only if the desired prescale value is '000' or '001'.

To change prescaler from the WDT to the TMR0 module use the sequence shown in Example 7-2. This precaution must be taken even if the WDT is disabled.

EXAMPLE 7-2: CHANGING PRESCALER (WDT→TMR0)

```
1. CLRWDT                  ;Clear WDT and
                           ;prescaler
2. BSF STATUS, RP0
3. MOVLW B'xxxx0xxx'       ;Select TMR0, new
                           ;prescale value and
                           ;clock source
4. MOVWF OPTION            ;
5. BCF STATUS, RP0
```

TABLE 7-1: SUMMARY OF TMR0 REGISTERS

Register Name	Function	Address	Power-on Reset Value
TMR0	Timer/counter register	01h	xxxx xxxx
OPTION	Configuration and prescaler assignment bits for TMR0. See Figure 4-4	81h	1111 1111
INTCON	TMR0 overflow interrupt flag and mask bits See Figure 4-5	0Bh	0000 000x

TABLE 7-2: REGISTERS ASSOCIATED WITH TMR0

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
01	TMR0	TIMER0							
0B/8B	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
81	OPTION	RBPU	INTEDG	RTS	RTE	PSA	PS2	PS1	PS0
85	TRISA	—	—	TRISA 5	TRISA 4	TRISA 3	TRISA 2	TRISA 1	TRISA 0

Legend — = Unimplemented locations, Read as '0'
Shaded boxes are not used by TMR0 module.

8.0 TIMER1 (TMR1) MODULE

Timer1 is a 16-bit timer/counter consisting of two 8-bit registers (TMR1H and TMR1L) which are readable and writable. TMR1 increments from 0000h to FFFFh and rolls over to 0000h. An interrupt, if enabled, is generated on overflow which is latched in interrupt flag bit TMR1IF (PIR1<0>). This interrupt can be enabled or disabled using the timer1 interrupt enable bit TMR1IE (PIE1<0>).

TMR1 can operate in one of two modes:

- As a timer
- As a counter

This is determined by the clock select bit, TMR1CS (TICON<1>).

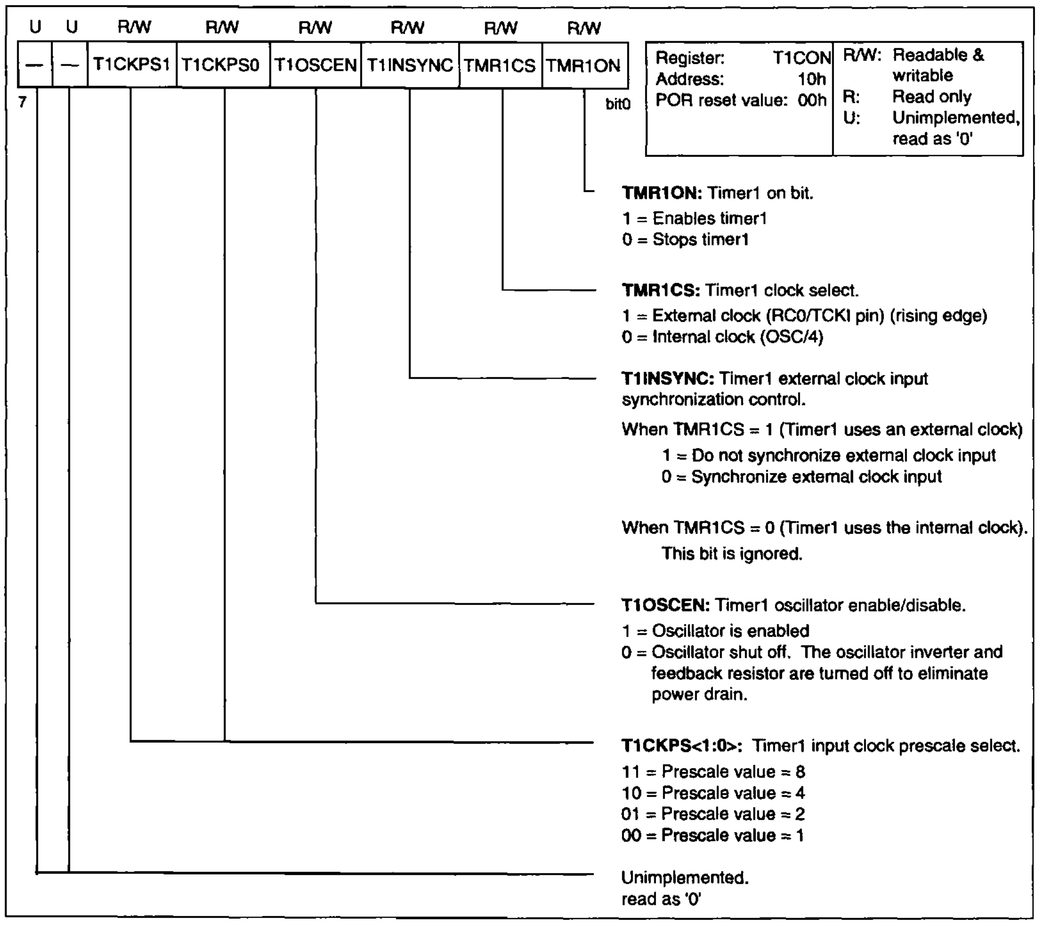
In timer mode, TMR1 increments every instruction cycle. In counter mode, it increments on every rising edge of external clock input on pin RC0/T1CKI.

Timer1 can be turned on or off using the control bit TMR1ON (T1CON<0>). See description of T1CON control register for all control bits related to Timer1.

Timer1 also has an internal "reset input". This reset can be generated by CCP1 (Capture/compare/PWM) module. See Section 10 for details. Figure 8-1 shows the Timer1 control register.

When the Timer1 oscillator is enabled (T1OSCEN is set), the RC1/T1CKO pin becomes an output. That is, the TRISC<1> value is ignored. The RC0/T1CKI pin should normally be configured as an input (for external clock). However, this pin can be configured as an output, if self-clocking (through the output pin) is desired.

FIGURE 8-1: T1CON: TIMER1 CONTROL REGISTER



U	U	R/W	R/W	R/W	R/W	R/W	R/W
—	—	T1CKPS1	T1CKPS0	T1OSCEN	T1INSYNC	TMR1CS	TMR1ON
7							bit0

Register: T1CON Address: 10h POR reset value: 00h	R/W: Readable & writable R: Read only U: Unimplemented, read as '0'

TMR1ON: Timer1 on bit.
1 = Enables timer1
0 = Stops timer1

TMR1CS: Timer1 clock select.
1 = External clock (RC0/TCKI pin) (rising edge)
0 = Internal clock (OSC/4)

T1INSYNC: Timer1 external clock input synchronization control.
When TMR1CS = 1 (Timer1 uses an external clock)
1 = Do not synchronize external clock input
0 = Synchronize external clock input

When TMR1CS = 0 (Timer1 uses the internal clock).
This bit is ignored.

T1OSCEN: Timer1 oscillator enable/disable.
1 = Oscillator is enabled
0 = Oscillator shut off. The oscillator inverter and feedback resistor are turned off to eliminate power drain.

T1CKPS<1:0>: Timer1 input clock prescale select.
11 = Prescale value = 8
10 = Prescale value = 4
01 = Prescale value = 2
00 = Prescale value = 1

Unimplemented.
read as '0'

8.1 TMR1 Operation in Timer Mode

Timer mode is selected by clearing the TMR1CS (T1CON<1>) bit. In this mode, the input clock to the timer is OSC/4. The synchronize control bit T1INSYNC (T1CON<2>) has no effect since the internal clock is always in sync.

8.2 TMR1 Operation in Synchronized Counter Mode

Counter mode is selected by setting the TMR1CS bit. In this mode the timer increments on every rising edge of clock input on pin RC0/T1OSCO/T1CKI.

If T1INSYNC is cleared, then the external clock input is synchronized with internal phase clocks. The synchronization is done after the prescaler stage. The prescaler stage is an asynchronous ripple counter.

In this configuration, during SLEEP mode, TMR1 will not increment even if external clock is present, since the synchronization circuit is shut off. The prescaler however will continue to increment.

8.2.1 EXTERNAL CLOCK INPUT TIMING FOR SYNCHRONIZED COUNTER MODE

In synchronized counter mode, the prescaler output is sampled twice every instruction cycle. Therefore, the following restrictions apply to external clock input:

when prescaler is 1:1:

T_{TCKIH} = T1CKI high time $\geq 2\,tosc + \Delta T$

T_{TCKIL} = T1CKI low time $\geq 2\,tosc + \Delta T$

where ΔT = small RC delay (see timing specifications)

when prescaler is higher:

The input clock is divided by two or more, so the prescaler output is symmetrical. The requirements are then:

Prescaler out high time = Prescaler out low time

$$= \frac{N \bullet T_{T1CKIP}}{2} \geq 2\,tosc + \Delta T$$

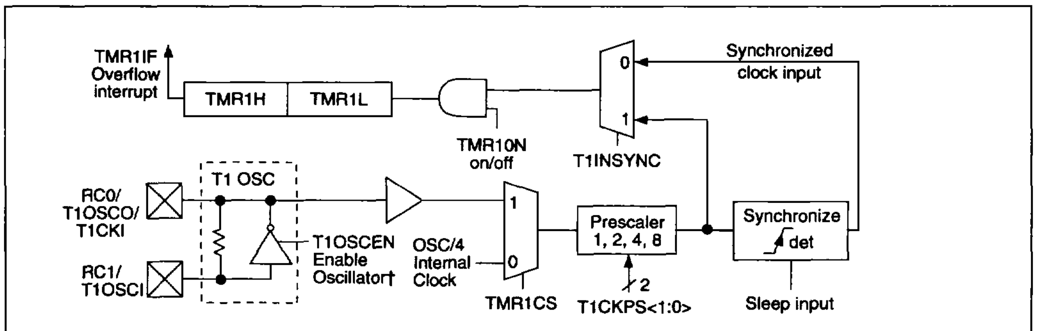
$$\text{or } T_{T1CKIP} \geq \frac{4\,tosc + \Delta T}{N}$$

where	TT1CKIP =	T1CKI input clock period
	N =	prescale value (2, 4, 8)
	ΔT =	small RC delay (see timing specifications)

Note that no requirement on minimum high time or low time is mentioned. However, if the pulse is too small it may not be recognized. Hence a minimum high and low time is specified.

$T_{T1CKIH} \geq \Delta T$ (see timing specification #45)
$T_{T1CKIL} > \Delta T$ (see timing specification # 46)

FIGURE 8-2: TMR1 BLOCK DIAGRAM



† When the T1 OSCEN bit is cleared, the inverter and feedback resistor are turned off. This eliminates power drain.

8.3 TMR1 Operation in Asynchronous Counter Mode

If the control bit T1INSYNC is set, the external clock input is not synchronized. The timer continues to increment asynchronous to the internal phase clocks. The timer will continue to run during SLEEP and generate an interrupt on overflow which will wake up the processor. However, special precautions in software are needed to read/write the timer (see Section 8.3.2).

In asynchronous counter mode, Timer1 can not be used as timebase for capture or compare operations.

8.3.1 EXTERNAL CLOCK INPUT TIMING WITH UNSYNCHRONIZED CLOCK

If T1INSYNC is set, the timer will increment completely asynchronously. The input clock must meet a certain minimum high time and low time requirements, as specified in timing specifications #45 and #46.

8.3.2 READING AND WRITING TIMER1 IN ASYNCHRONOUS COUNTER MODE

Reading TMR1H or TMR1L while the timer is running off external asynchronous clock will guarantee a valid read (taken care of in hardware). However, the user should keep in mind that reading the 16-bit timer in two 8-bit values itself poses certain problems since the timer may overflow between the reads.

For writes, it is recommended that the user simply stop the timer and write the desired values. A write contention may occur by writing to the timer registers while the register is incrementing. This may produce an unpredictable value in the timer register.

Writing to the timer is no problem, if the clock is slower than the instruction cycle time. Reading the 16-bit value requires some care. Example 8-1 is an example routine to read the 16-bit timer value. This is useful if the timer cannot be stopped.

EXAMPLE 8-1: READING A 16-BIT FREE-RUNNING TIMER

```
;  All Interrupts are disabled
    MOVF    TMR1H, Wreg     ; Read high byte
    MOVWF   TMPH            ;
    MOVF    TMR1L, Wreg     ; Read low byte
    MOVWF   TMPL            ;
    MOVF    TMR1H, Wreg     ; Read high byte
    SUBWF   TMPH, Wreg      ; Sub 1st read
                            ;   with 2nd read
    BTFSC   STATUS,  Z      ;   is result = 0
    GOTO    CONTINUE        ; Good 16-bit read
;
; TMR1L may have rolled over between the read
; of the high and low bytes. Reading the high
; and low bytes now will read a good value.
;
    MOVF    TMR1H, Wreg     ; Read high byte
    MOVWF   TMPH            ;
    MOVF    TMR1L, Wreg     ; Read low byte
    MOVWF   TMPL            ;
;  Re-enable Interrupt (if required)
CONTINUE    :               ; Continue with your
                            ; code
```

8.4 Timer1 Oscillator

A crystal oscillator circuit is built in between T1CKI pin (input) and T1CKO (amplifier output). It is enabled by setting control bit T1OSCEN (T1CON<3>). The oscillator is a low-power oscillator rated up to 200KHz. It will continue to run during SLEEP. It is primarily intended for a 32KHz crystal. Table 8-1 shows the capacitor selection for the Timer1 oscillator.

The Timer1 oscillator is identical to the LP oscillator. The user must allow software time-out to ensure proper oscillator start-up.

TABLE 8-1: CAPACITOR SELECTION FOR THE TIMER1 OSCILLATOR

Osc Type	Freq	C1	C2
LP	32 KHz§	15 pF	15 pF
	100 KHz	15 pF	15 pF
	200 KHz	0 - 15 pF	0 - 15 pF
Higher capacitance increases the stability of oscillator but also increases the start-up time. These values are for design guidance only. §For $V_{DD} > 4.5V$, C1 = C2 ~ 30pf is recommended.			

8.5 Resetting Timer1 using CCP1 Trigger Output

If CCP1 module is configured in compare mode to generate a "special event" trigger (CCP1M<3:0> = 1011), this signal will reset timer1.

Timer1 must be configured for timer or synchronized counter mode operation to take advantage of this feature. If the Timer1 is running in asynchronous counter mode, this reset operation may not work.

In the event that a write to Timer1 coincides with a reset trigger from CCP1, the write will take precedence.

In this mode of operation, the CCPR1H:CCPR1L registers pair effectively becomes the period register for the timer1.

8.6 Resetting of Timer Registers

TMR1H and TMR1L registers are not reset on POR or any other reset except by the CCP1 special reset trigger.

T1CON register is reset to 00h on Power-On Reset. In any other reset, the register is unaffected.

8.7 Prescaler

The prescaler counter is cleared on writes to the TMR1H or TMR1L registers.

TABLE 8-2: REGISTERS ASSOCIATED WITH TIMER1 AS A TIMER/COUNTER

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0B/8B	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
0C	PIR1	PSPIF	—	—	—	SSPIF	CCP1IF	TMR2IF	TMR1IF
8C	PIE1	PSPIE	—	—	—	SSPIE	CCP1IE	TMR2IE	TMR1IE
0E	TMR1L	Timer1 Least Significant Byte							
0F	TMR1H	Timer1 Most Significant Byte							
10	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	T1INSYNC	TMR1CS	TMR1ON

Legend — = Unimplemented locations, Read as '0'
Note: Shaded boxes are not used by Timer1 module.

9.0 TIMER2 (TMR2) MODULE

Timer2 is an 8-bit timer with a prescaler and a postscaler. It is especially suitable as PWM time-base (for PWM mode of CCP module). TMR2 is a readable and writable register.

The input clock (osc/4) has a prescale option of 1, 4 or 16 (selected by control bits TCKPS1, TCKPS0, register T2CON).

Timer2 has an 8-bit period register PR2. Timer2 increments from 00h until it matches PR2 and then resets t0 00h. PR2 is a readable and writable register.

The overflow (or match) output of TMR2 goes through a 4-bit postscaler (which gives a 1:1 to 1:16 scaling) to generate timer2 interrupt (latched in TMR2IF bit, PIR<1>).

Timer2 can be shut off using TMR2ON (T2CON<2>) control bit to minimize power consumption.

Figure 9-2 shows the Timer2 control register.

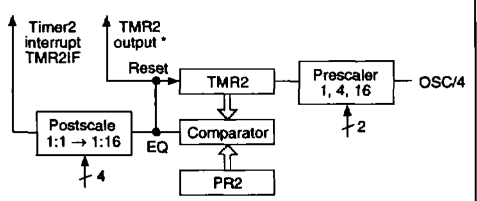
9.1 Prescaler and Postscaler

The prescaler and postscaler counters are cleared when any of the following occurs: a write to the TMR2 register, a write to the T2CON register, or any device reset (Power-On Reset, $\overline{\text{MCLR}}$ reset, or Watchdog Timer reset).

9.2 Output of TMR2

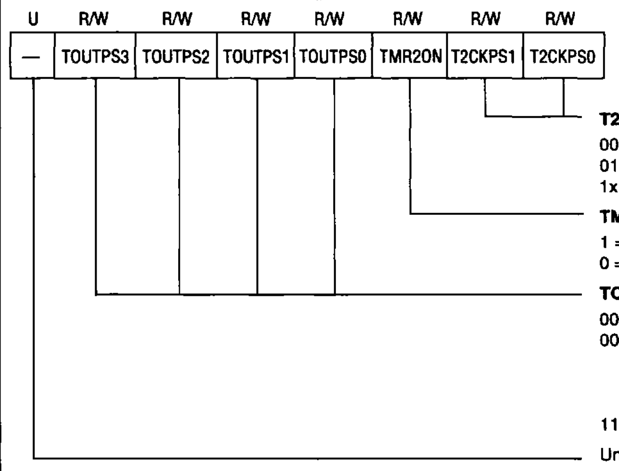
The output of TMR2 (before the postscaler) is fed to the synchronous serial port module which optionally uses it to generate shift clock.

FIGURE 9-1: TIMER2 BLOCK DIAGRAM



* TMR2 output can be software selected by the SSP module as baud clock.

FIGURE 9-2: T2CON: TIMER2 CONTROL REGISTER



U R/W R/W R/W R/W R/W R/W R/W
— TOUTPS3 TOUTPS2 TOUTPS1 TOUTPS0 TMR2ON T2CKPS1 T2CKPS0

Register: T2CON	R/W: Readable & writable
Address: 12h	
Reset value: 00h	U: Unimplemented

T2CKPS<1:0>: Timer2 clock prescaler select.
00 = Prescaler is 1
01 = Prescaler is 4
1x = Prescaler is 16

TMR2ON: Timer2 on/off control.
1 = TMR2 is on
0 = TMR2 is off

TOUTPS<3:0>: Timer2 output postscale select
0000 = Postscaler is 1
0001 = Postscaler is 2
•
•
•
1111 = Postscaler is 16

Unimplemented.
Read as '0'

TABLE 9-1: REGISTERS ASSOCIATED WITH TIMER2 AS A TIMER/COUNTER

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0B	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
0C	PIR1	PSPIF	—	—	—	SSPIF	CCP1IF	TMR2IF	TMR1IF
8C	PIE1	PSPIE	—	—	—	SSPIE	CCP1IE	TMR2IE	TMR1IE
11	TMR2	Timer2							
12	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
92	PR2	Timer2 period Register							

Legend — = Unimplemented locations, Read as '0'
Note: Shaded boxes are not used by Timer2 module.

10.0 CAPTURE/COMPARE/PWM MODULE

The PIC16C64 has a Capture/Compare/PWM (CCP) module consisting of a 16-bit register which can operate as a 16-bit capture register, as a 16-bit compare register or as a PWM output.

CCP1 module:

Capture/compare/PWM register1 (CCPR1) is made up of two 8-bit sections: low byte, CCPR1L and high byte, CCPR1H. Both are readable and writable.

10.1 Capture Mode

In Capture mode, CCPR1 captures the 16-bit value of TMR1 when an event occurs on pin RC2/CCP1. An event is defined as:

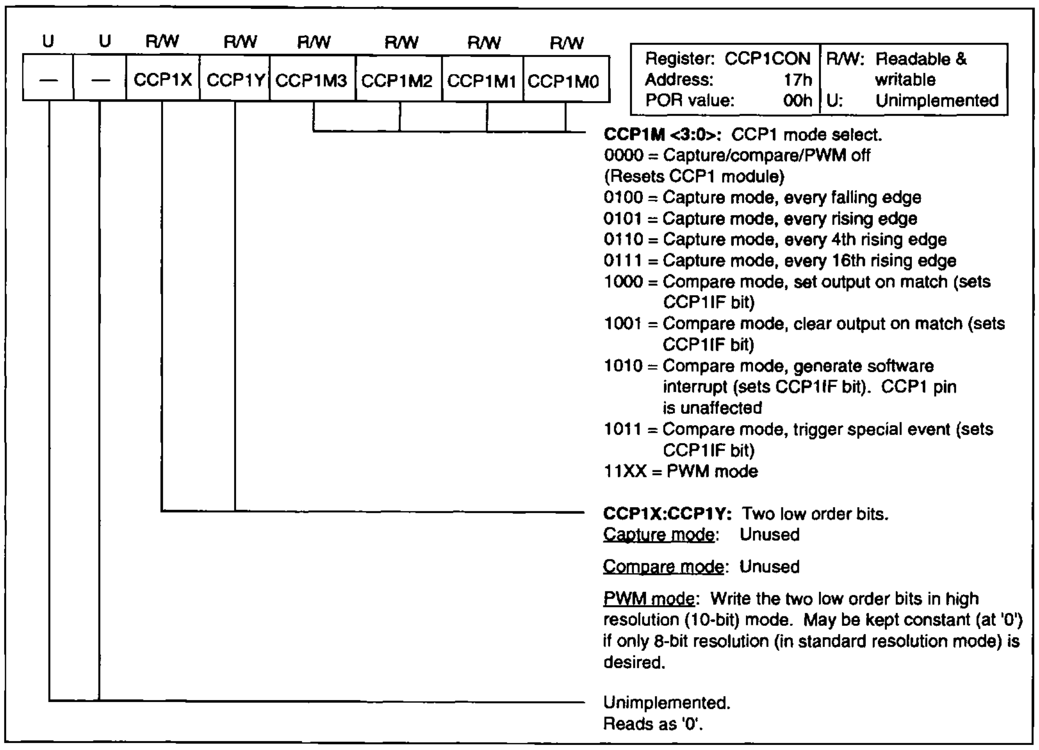
1. A falling edge
2. A rising edge
3. Every 4 rising edges
4. Every 16 rising edges

One of these is selected by the control bits CCP1M <3:0> in register CCP1CON. When a capture is made the interrupt request flag, CCP1IR bit (PIR<2>) is set. It must be reset in software. If another capture occurs before the value in register CCPR1 is read, the old captured value will be lost. In capture mode, the RC2/CCP1 pin should be configured as an input through its corresponding TRIS bit.

Note: If the RC2/CCP1 is configured as an output, a write to the port can cause a capture condition.

When the capture mode is changed, a false capture interrupt may be generated. The user should clear the CCP1IF bit following any such change in operating mode.

FIGURE 10-1: CCP1CON REGISTER



U	U	R/W	R/W	R/W	R/W	R/W	R/W
—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0

Register: CCP1CON	R/W: Readable & writable
Address: 17h	
POR value: 00h	U: Unimplemented

CCP1M <3:0>: CCP1 mode select.
0000 = Capture/compare/PWM off (Resets CCP1 module)
0100 = Capture mode, every falling edge
0101 = Capture mode, every rising edge
0110 = Capture mode, every 4th rising edge
0111 = Capture mode, every 16th rising edge
1000 = Compare mode, set output on match (sets CCP1IF bit)
1001 = Compare mode, clear output on match (sets CCP1IF bit)
1010 = Compare mode, generate software interrupt (sets CCP1IF bit). CCP1 pin is unaffected
1011 = Compare mode, trigger special event (sets CCP1IF bit)
11XX = PWM mode

CCP1X:CCP1Y: Two low order bits.
Capture mode: Unused

Compare mode: Unused

PWM mode: Write the two low order bits in high resolution (10-bit) mode. May be kept constant (at '0') if only 8-bit resolution (in standard resolution mode) is desired.

Unimplemented.
Reads as '0'.

10.1.1 PRESCALER DETAILS

There are four prescaler setting, specified by the CCP1M3-CCP1M0 bits. Whenever the CCP module is turned off, or the CCP module is not in capture mode, the prescaler counter is cleared. This means that any reset will clear the prescaler counter.

Switching from one capture prescaler to another may generate an interrupt. Also, the prescaler counter will not be cleared, and therefore the first capture may be from a non-zero prescaler. Example 10-1 shows the recommended way to switch between capture prescalers. This example also clears the prescaler counter and will not generate the "false" interrupt.

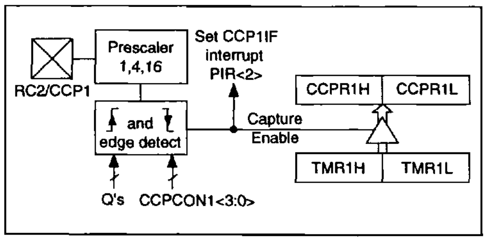
EXAMPLE 10-1: CHANGING BETWEEN CAPTURE PRESCALERS

```
CLRF    CCP1CON       ; Turn CCP module off
MOVLW   NEW_CAPT_PS   ; Load the W reg with
                      ;  the new prescaler
                      ;  mode value
MOVWF   CCP1CON       ; Load CCP1CON with
                      ;  this value
```

10.1.2 TIMER1 MODE SELECTION

Timer1 must be running in timer mode or synchronized counter mode for the CCP1 modules to use the capture feature. In asynchronous counter mode the capture operation may not work.

FIGURE 10-2: CAPTURE MODE OPERATION



10.2 Compare Mode

In compare mode, the 16-bit CCPR1 register value is constantly compared against the timer1. When a match occurs, the RC2/CCP1 pin is:

- Driven High
- Driven Low
- Remains Unchanged

The action on the pin is based on the control bits CCP1M <3:0> in register CCP1CON. At the same time, a compare interrupt is also generated (bit CCP1IR, register CCP1CON). The user must set the RC2/CCP1 pin as an output through the TRISC<2> bit.

Note: Clearing the CCP1CON register will force the RC2/CCP1 compare output latch to the default low level. This is not the data latch.

10.2.1 TIMER1 MODE SELECTION

Timer1 must be running in timer mode or synchronized counter mode if the CCP1 module is using the compare feature. In asynchronous counter mode the compare operation may not work.

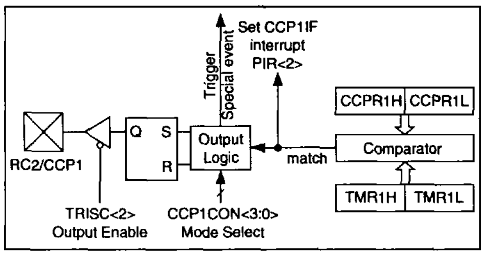
10.2.2 SOFTWARE INTERRUPT MODE

Another compare mode is software interrupt mode in which the CCP1 pin is not affected. Only CCP1IF interrupt is generated.

10.2.3 SPECIAL TRIGGER

The special trigger output of CCP1 is an internal Hardware Trigger which can be used to reset the timer1. This allows the CCPR1 register to effectively be a 16-bit programmable period register for timer1.

FIGURE 10-3: COMPARE MODE OPERATION



10.3 PWM Mode

In Pulse Width Modulation (PWM) mode, the RC2/CCP1 produces up to 10-bit resolution PWM output. This pin must be configured as an output through the TRISC<2> bit. In PWM mode, the user writes the 8-bit duty-cycle value to the low byte of the CCPR1 register, namely CCPR1L. The high-byte, CCPR1H is used as the slave buffer to the low byte. The 8-bit data is transferred from the master to the slave when the PWM1 output is set (i.e. at the beginning of the duty cycle). This double buffering is essential for glitchless PWM output. In PWM mode CCPR1H is readable but not writable. The period of the PWM is determined by timer2 period register (PR2).

PWM period is =
[(PR2) + 1] • 4 tosc • (TMR2 prescale value)

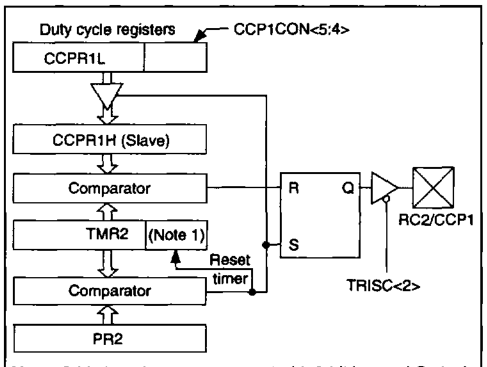
PWM duty cycle =
(DC1) • tosc • (TMR2 prescale value)

where DC1 = 10 bit value from CCPR1L and CCP1CON<5:4> concantenated.

The PWM output resolution is therefore programmable up to a maximum of 10-bit.

Note: Clearing the CCP1CON register will force the RC2/CCP1 PWM output latch to the default low level. This is not the I/O data latch.

FIGURE 10-4: SIMPLIFIED PWM BLOCK DIAGRAM



Note: 8-bit timer is concantenated with 2-bit internal Q clock or 2 bits of the prescaler to create 10-bit time base.

TABLE 10-1: PWM FREQUENCY VS RESOLUTION AT 20 MHZ

Max Resolution (High Resolution Mode)	Frequency		
	TMR2 Prescale = 1	TMR2 Prescale = 4	TMR2 Prescale = 16
10 bit	19.53 KHz	4.88 KHz	1.22 KHz
9 bit	39.06 KHz	9.77 KHz	2.44 KHz
8 bit	78.13 KHz	19.53 KHz	4.88 KHz

TABLE 10-2: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS AT 20 MHz

PWM Frequency	1.22 KHz	4.88 KHz	19.53 KHz	78.13 KHz	157.5 KHz	210.53 KHz
Timer Prescaler (1, 4, 16)	16	4	1	1	1	1
PR2 Value	0xFF	0xFF	0xFF	0xFF	0x7F	0x5F
Resoution (High-resolution mode†)	10-bit	10-bit	10-bit	8-bit	7-bit	6.5-bit
Resolution (Standard-resolution mode†)	8-bit	8-bit	8-bit	6-bit	5-bit	4.5-bit

† Standard resolution mode has the CCPIX:CCPIY bit constant (or '0'), and only compares the TMR2 against the PR2. The Q-cycles are not used.

TABLE 10-3: REGISTERS ASSOCIATED WITH TIMER1 AND CAPTURE

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0B/8B	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
0C	PIR1	PSPIF	—	—	—	SSPIF	CCP1IF	TMR2IF	TMR1IF
8C	PIE1	PSPIE	—	—	—	SSPIE	CCP1IE	TMR2IE	TMR1IE
0E	TMR1L	Timer1 Least Significant Byte							
0F	TMR1H	Timer1 Most Significant Byte							
10	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	T1INSYNC	TMR1CS	TMR1ON
15	CCPR1L	Timer1 Capture Register (LSB)							
16	CCPR1H	Timer1 Capture Register (MSB)							
17	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0

Legend — = Unimplemented locations, Read as '0' Note: Shaded boxes are not used in this mode.

TABLE 10-4: REGISTERS ASSOCIATED WITH TIMER1 AND COMPARE

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0B/8B	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
0C	PIR1	PSPIF	—	—	—	SSPIF	CCP1IF	TMR2IF	TMR1IF
8C	PIE1	PSPIE	—	—	—	SSPIE	CCP1IE	TMR2IE	TMR1IE
0E	TMR1L	Timer1 Least Significant Byte							
0F	TMR1H	Timer1 Most Significant Byte							
10	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	T1INSYNC	TMR1CS	TMR1ON
15	CCPR1L	Timer1 Compare Register (LSB)							
16	CCPR1H	Timer1 Compare Register (MSB)							
17	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0

Legend — = Unimplemented locations, Read as '0' Note: Shaded boxes are not used in this mode.

TABLE 10-5: REGISTERS ASSOCIATED WITH TIMER2 AND PWM

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0B/8B	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
0C	PIR1	PSPIF	—	—	—	SSPIF	CCP1IF	TMR2IF	TMR1IF
8C	PIE1	PSPIE	—	—	—	SSPIE	CCP1IE	TMR2IE	TMR1IE
11	TMR2	Timer2							
92	PR2	Timer2 period Register							
12	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
15	CCPR1L	Timer2 Duty Cycle Register							
16	CCPR1H	Timer2 Duty Cycle Register (Slave)							
17	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0

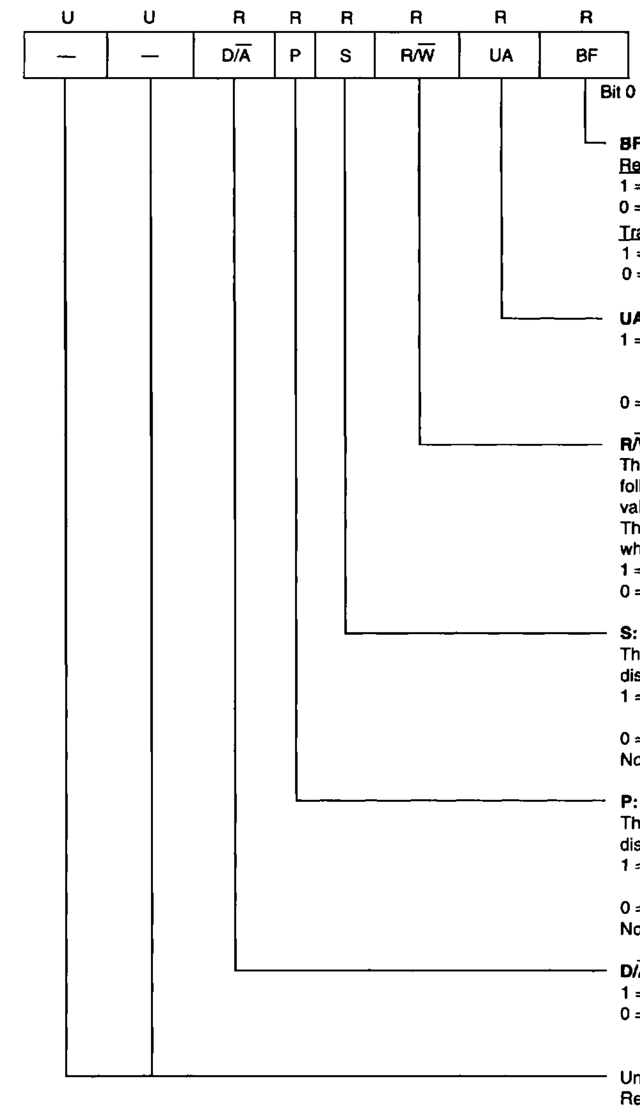
Legend — = Unimplemented locations, Read as '0' Note: Shaded boxes are not used in this mode.

11.0 SYNCHRONOUS SERIAL PORT (SSP)

The Synchronous Serial Port (SSP) 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 SSP module can operate in one of two modes:

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

FIGURE 11-1: SSPSTAT: SYNC SERIAL PORT STATUS REGISTER



Register SSPSTAT Addr: 94h POR value: 00h	R/W: Readable & writable bit R: Read only bit U: Unimplemented, read as '0'

BF: Buffer full
Receive Mode
1 = Receive complete, SSPBUF is full
0 = Receive not complete, SSPBUF is empty
Transmit Mode
1 = Transmit in progress, SSPBUF is full
0 = Transmit complete, SSPBUF is empty

UA: Update Address (10-bit I^2C slave mode only).
1 = Indicate that the user needs to update the address in the SSPADD register. See Section 11.2 for details.
0 = Address does not need to be updated.

R/$\overline{W}$: Read/write bit information (I^2C mode only)
This bit holds the R/$\overline{W}$ bit information received following the last address match. This bit is only valid during the transmission.
The user may use this bit in software to determine whether transmission or reception is in progress.
1 = Read
0 = Write

S: Start bit (I^2C mode only) Flag
This bit is cleared when the SSP module is disabled (SSPEN is cleared)
1 = Indicates that a start bit has been detected last. This bit is 0 on reset.
0 = Start bit was not detected last
Note: This bit also operates in the Master Modes

P: Stop bit (I^2C mode only) Flag
This bit is cleared when the SSP module is disabled (SSPEN is cleared)
1 = Indicates that a stop bit has been detected last.
0 = Stop bit was not detected last
Note: This bit also operates in the Master Modes

D/$\overline{A}$: Data/Address bit (I^2C mode only)
1 = Indicates that the last byte received was data
0 = Indicates that the last byte received was address

Unimplemented.
Reads as '0'.

FIGURE 11-2: SSPCON: SYNC SERIAL PORT CONTROL REGISTER

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	Bit 0

Register SSPCON Addr: 14h POR value: 00h	R/W: Readable & writable bit R: Read only bit U: Unimplemented, read as '0'

SSPM<3:0>: Synchronous serial port mode select
0000 = SPI master mode, clock = osc/4
0001 = SPI master mode, clock = osc/16
0010 = SPI master mode, clock = osc/64
0011 = SPI master mode, clock = (TMR2 output/2)
0100 = SPI slave mode, clock = SCK pin. $\overline{SS}$ pin control enabled.
0101 = SPI slave mode, clock = SCK pin. $\overline{SS}$ pin control disabled. $\overline{SS}$ can be used as I/O pin.
0110 = I²C slave mode, 7-bit address
0111 = I²C slave mode, 10-bit address
1011 = I²C master mode support enabled (slave idle)
1110 = I²C slave mode, 7-bit address with master mode support enabled
1111 = I²C slave mode, 10-bit address with master mode support enabled

CKP: Clock polarity select.
In SPI modes:
1 = Transmit happens on falling edge, receive on rising edge. Idle state for clock is a High level.
0 = Transmit happens on rising edge, receive on falling edge. Idle state for clock is a Low level.
In I²C modes:
SCK release control
1 = Enable clock
0 = Holds clock low (clock stretch)
Note: Used to ensure data setup time

SSPEN: Sync serial port enable
In SPI modes:
1 = Enables serial port and configures SCK, SDO and SDI as serial port pins.
0 = Disables serial port and configures these pins as I/O port pins.
In I²C modes:
1 = Enables the serial port and configures SDA and SCL pins as serial port pins.
0 = Disables serial port and configures these pins as I/O port pins.

In both modes, when enabled, these pins must be properly configured as input or output.

SSPOV: Receive overflow flag.
In SPI modes:
1 = A new byte is received while 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. The user must read the SSPBUF, even if only transmitting data, to avoid setting overflow. In master mode overflow bit is not set since each new reception (and transmission) is initiated by writing to SSPBUF.
In I²C modes:
1 = A byte is received while the SSPBUF is still holding the previous byte.
SSPOV is a don't care in transmit mode.

SSPOV must be cleared in software in either mode.

WCOL: Write collision detect.
1 = the SSPBUF register is written while it is still transmitting the previous word.
Must be cleared in software.
0 = No collision

11.1 SPI Mode

The SPI mode allows 8-bits of data to be synchronously transmitted and received simultaneously. To accomplish communication, typically three pins are used:

- Serial Data Out (SDO)
- Serial Data In (SDI)
- Serial Clock (SCK)

Additionally a fourth pin may be used when in a slave mode of operation:

- Slave Select ($\overline{SS}$)

When initializing the SPI, several options need to be specified. This is done by appropriately programming the control bit in the SSPCON register (SSPCON<5:0>). These control bits allow the following to be specified:

- Master Mode (SCK is the clock output)
- Slave Mode (SCK is the clock input)
- Clock Polarity (Output/Input data on the Rising/ Falling edge of SCK)
- Clock Rate(Master mode only)
- Slave Select Mode (Slave mode only)

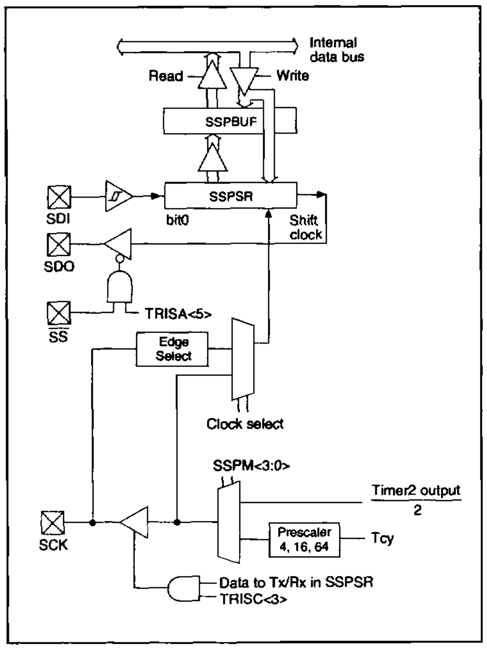
The SSP consists of a transmit/receive Shift Register (SSPSR) and a Buffer register (SSPBUF). The SSPSR shifts the data in and out of the device, while the SSPBUF holds the data that was written to the SSPSR, until the received data is ready. Once the 8-bits of data have been received, that information is moved to the SSPBUF register, the Buffer Full (BF) bit (SSPSTAT <0>) and the SSPIF bit are set. This double buffering of the received data (SSPBUF), allows the next byte to start reception before reading the data that was received. Any write to the SSPBUF register during transmission/reception of data will be ignored, and the write collision detect (WCOL) bit (SSPCON<7>) will be set. User software must clear the WCOL bit so that it can be determined if the following write(s) to the SSPBUF completed successfully. When the application software is expecting to receive valid data, the SPPBUF should be read before the next byte of data to transfer is written to the SSPBUF. The Buffer Full (BF) bit (SSPSTAT<0>) indicates when the SSPBUF has been loaded with the received data (transmission is complete). When the SSPBUF is read, the BF bit is cleared. This data may be irrelevant if the SPI is only a transmitter. Generally the SSP Interrupt is used to determine when the transmission/reception has completed. The SSPBUF can then be read (if data is meaningful) and/or the SSPBUF (SSPSR) can be 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 11-1 shows the loading of the SSPBUF (SSPSR) for data transmission. The instructions with the comment fields beginning with ;*** are only required if the received data is meaningful.

EXAMPLE 11-1: LOADING THE SSPBUF (SSPSR) REGISTER

```
LOOP BSF     STATUS, RP0   ;Specify Bank 1
     BTFSS   SSPSTAT, BF   ;Has data been
                           ;received
                           ;(transmit complete)?
     GOTO    LOOP          ;No
     BCF     STATUS, RP0   ;Specify Bank 0
     MOVF    SSPBUF, W     ;W reg = contents
                           ;of SSPBUF
     MOVWF   RXDATA        ;*** Save in user RAM
     MOVF    TXDATA, W     ;W reg = contents of
                           ;TXDATA
     MOVWF   SSPBUF        ;New data to xmit.
```

The block diagram of the SSP module, when in SPI mode (Figure 11-3), shows that the SSPSR is not directly readable or writable, and can only be accessed from addressing the SSPBUF register. Additionally, the SSP status register (SSPSTAT) indicates the various status conditions.

FIGURE 11-3: SSP BLOCK DIAGRAM (SPI MODE)



To enable the serial port, the SSP enable bit (SSPEN) must be set. This configures the SDI, SDO, SCK, and SS pins as serial port pins. For the pins to behave as the serial port function, they must have their data direction bits (in the TRISC register) appropriately programmed. That is:

- SDI must have TRISC<4> set
- SDO must have TRISC<5> cleared
- SCK (Master mode) must have TRISC<3> cleared
- SCK (Slave mode) must have TRISC<3> set
- $\overline{\text{SS}}$ must have TRISA<5> set

Any serial port function that is not desired may be overridden by programming the corresponding data direction (TRIS) register to the opposite value. An example would be in master mode where you are only sending data (to a display driver), then both SDI and SS could be used as general purpose inputs by clearing their corresponding TRIS register bits.

Figure 11-4 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

The master can initiate the data transfer at any time because it controls the SCK. How the master knows when the slave (Processor 2) wishes to broadcast data is determined by the software protocol

In master mode the data is transmitted/received as soon as the SSPBUF is written to. If the SPI is only going to receive, the SCK output could be disabled (programmed as an input). The SSPSR register will continue to shift in the signal present on the SDI pin at the programmed clock rate. As each byte is received, it will be loaded into the SSPBUF as if a normal received byte (interrupts and status bits appropriately set). This could be useful in receiver applications as a "line activity monitor" mode.

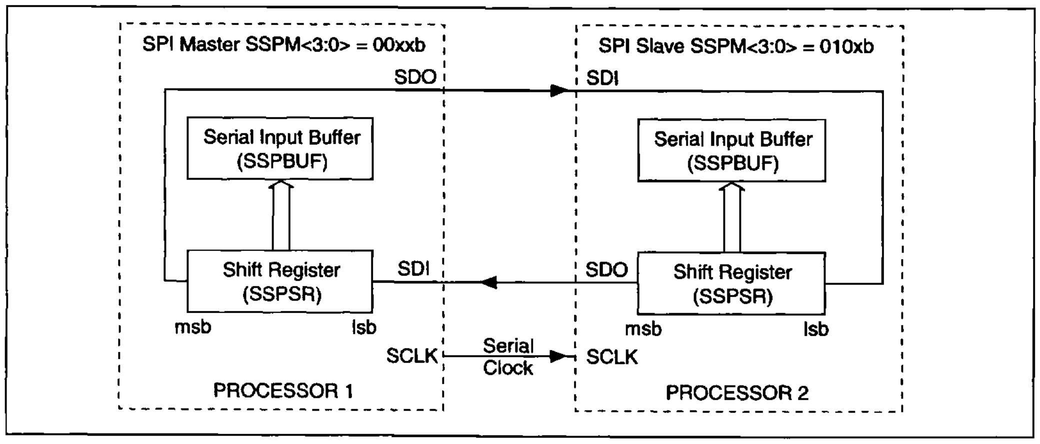
In slave mode, the data is transmitted and received as the external clock pulses appear on SCK. When the last bit is latched the interrupt flag (SSPIF) is set (PIR1<3>).

The clock polarity is selected by appropriately programming the CKP bit (SSPCON<4>). This then would give waveforms for SPI communication as shown in Figures 11-5 and 11-6. In master mode the SPI clock rate (bit rate) is user programmable to be one of the following:

- OSC / 4 (or Tcy)
- OSC / 16 (or 4 • Tcy)
- OSC / 64 (or 16 • Tcy)
- Timer2 output / 2

This allows a maximum bit clock frequency (at 20 MHz) of 5 MHz. When in slave mode the external clock must meet the minimum high and low times as specified in timing parameters 71 and 72.

FIGURE 11-4: SPI MASTER / SLAVE CONNECTION


SPI Master SSPM<3:0> = 00xxb
SDO
Serial Input Buffer (SSPBUF)
Shift Register (SSPSR)
SDI
msb
lsb
SCLK
PROCESSOR 1
Serial Clock
SPI Slave SSPM<3:0> = 010xb
SDI
Serial Input Buffer (SSPBUF)
SDO
Shift Register (SSPSR)
msb
lsb
SCLK
PROCESSOR 2

The $\overline{SS}$ pin allows a synchronous slave mode. The SPI must be in slave mode (SSPCON<3:0> = 04h) and the TRISA<5> bit must be set the for the synchronous slave mode to be enabled. When the $\overline{SS}$ pin is low, transmission and reception are enabled and the SDO pin is driven. When the $\overline{SS}$ pin goes high, the SDO pin is no longer driven, even if in the middle of a transmitted byte, and becomes a floating output. External pull-up / pull-down resisters may be desirable, depending on the application.

To emulate 2-wire communication, the SDO pin can be connected to the SDI pin. When the SPI needs to operate as a receiver the SDO pin can be configured as an input. This disables transmissions from the SDO. The SDI can always be left as an input (SDI function) since it cannot create a bus conflict.

FIGURE 11-5: SPI MODE TIMING (MASTER MODE OR SLAVE MODE W/O $\overline{SS}$ CONTROL)

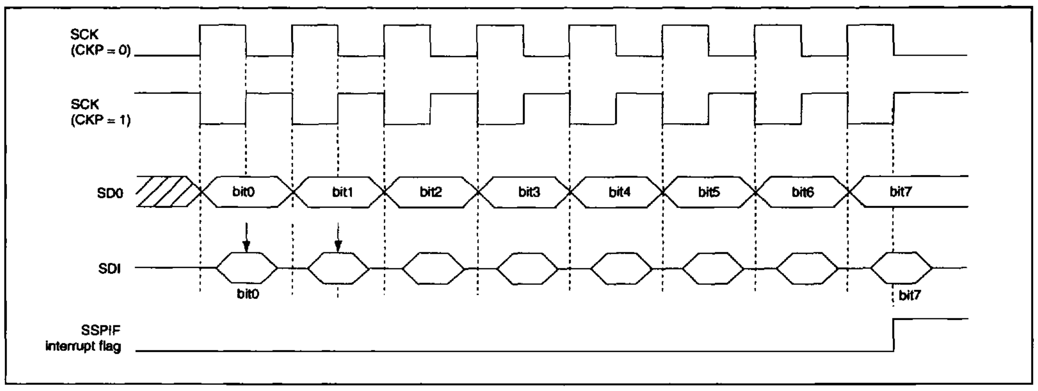


FIGURE 11-6: SPI MODE TIMING (SLAVE MODE WITH $\overline{SS}$ CONTROL)

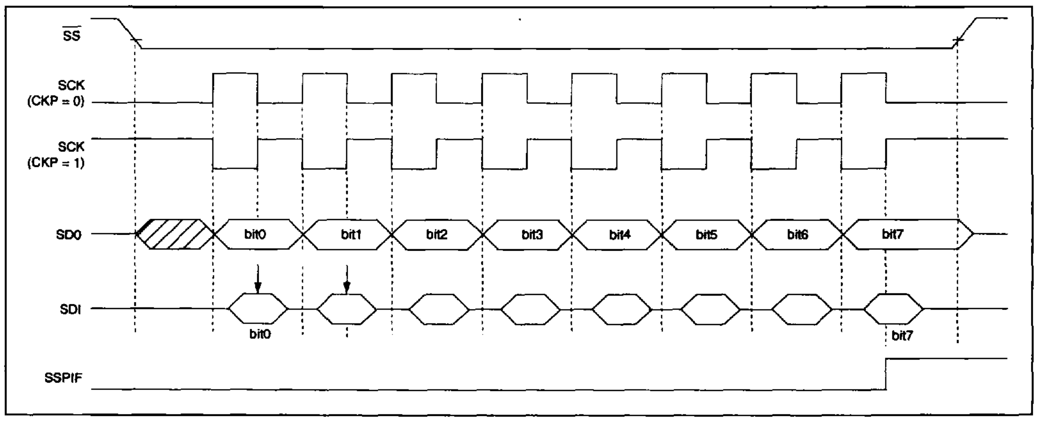


TABLE 11-1: REGISTERS ASSOCIATED WITH SPI OPERATION

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0B/8B	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
0C	PIR1	PSPIF	—	—	—	SSPIF	CCP1IF	TMR2IF	TMR1IF
8C	PIE1	PSPIE	—	—	—	SSPIE	CCP1IE	TMR2IE	TMR1IE
13	SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register							
14	SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
94	SSPSTAT	—	—	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF

Legend — = Unimplemented locations, Read as '0'   Note: Shaded boxes are not used by SSP module in SPI mode.

11.2 I²C Overview

This section gives an overview of the Inter-IC (I²C) bus, with Section 11.3 discussing the operation of the SSP module in I²C mode. The Inter-IC (I²C) bus is a two-wire serial interface developed by Philips/Signetics™. The original specification, or standard mode, was for data transfers of up to 100-Kbps. An enhanced specification, or fast mode, supports data transmission up to 400 Kbps. Both standard mode and fast mode devices will inter-operate if attached to the same bus.

The I²C interface employs a comprehensive protocol to ensure reliable transmission and reception of data. When transmitting data, one device is the "master" (generates the clock), while the other device(s) acts as the "slave". All Portions of the slave protocol are implemented in the SSP module's hardware, while portions of the master protocol will need to be addressed in the PIC16CXX software. Table 11-2 defines some of the I²C-bus terminology. For additional information on the I²C interface specification, please refer to the Philips/ Signetics™ document " The I²C-bus and how to use it". The order number for this document is 98-8080-575.

In the I²C interface protocol each device has an address. When a master wishes to initiate a data transfer, it first transmits the address of the device that it wishes to "talk" to. All devices "listen" to see if this is their address. Within this address, a bit specifies if the master wishes to read / write from / to the slave device. The master and slave are always in opposite modes (transmitter/receiver) of operation during a data transfer. That is they can be thought of operating in either of these two relations:

- Master-transmitter and Slave-receiver
- Slave-transmitter and Master-receiver

In both cases the master generates the clock signal.

The output stages of the clock (SCL) and data (SDA) lines must have an open-drain or open-collector in order to perform the wired-AND function of the bus. External pull-up resistors are used to ensure a high level, when no device is pulling the line down.. The number of devices that may be attached to the I²C bus is limited only by the maximum bus loading specification of 400 pF.

11.2.1 INITIATING AND TERMINATING DATA TRANSFER

During times of no data transfer (idle time), both the clock line (SCL) and the data line (SDA) are pulled high through the external pull-up resistors. The START and STOP conditions determine the start and stop of data transmission. The START condition is defined as a high to low transition of the SDA when the SCL is high. The STOP condition is defined as a low to high transition of the SDA when the SCL is high. Figure 11-7 shows the START and STOP conditions. The master generates these conditions for starting and terminating data transfer. Due to the definition of the START and STOP conditions, when data is being transmitted the SDA line can only change state when the SCL line is low.

FIGURE 11-7: START AND STOP CONDITIONS

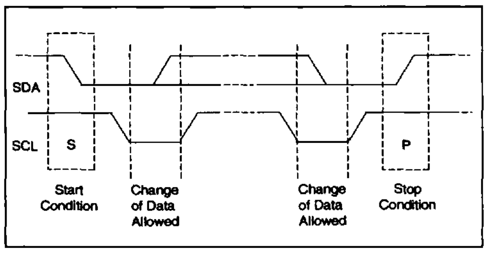


TABLE 11-2: I²C-BUS TERMINOLOGY

Term	Description
Transmitter	The device that sends the data to the bus
Receiver	The device that receives the data from the bus
Master	The device which initiates the transfer, generates the clock, and terminates the transfer
Slave	The device addressed by a master
Multi-master	More than one master device in a system. These masters can attempt to control the bus at the same time without corrupting the message
Arbitration	Procedure that ensures that only one of the master devices will control the bus. This ensures that the transfer data does not get corrupted.
Synchronization	Procedure where the clock signals of two or more devices are synchronized.

11.2.2 ADDRESSING I²C DEVICES

There are two address formats. The simplest is the 7-bit address format with a R/W bit (see Figure 11-8). The more complex is the 10-bit address with a R/W bit (see Figure 11-9). For 10-bit address format, two bytes must be transmitted with the first five bits specifying this to be a 10-bit address.

FIGURE 11-8: 7-BIT ADDRESS FORMAT

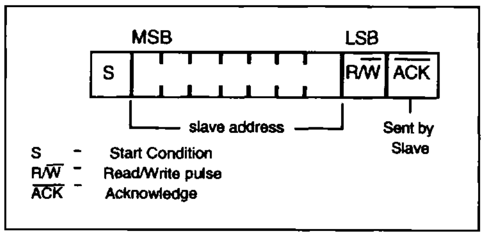
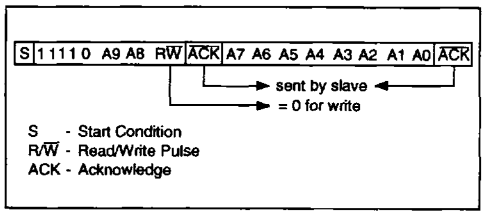


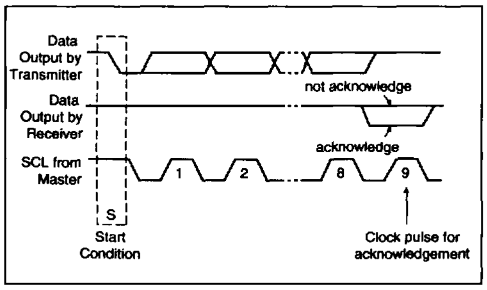
FIGURE 11-9: I²C 10-BIT ADDRESS FORMAT



11.2.3 TRANSFER ACKNOWLEDGE

All data must be transmitted per byte, with no limit to the number of bytes transmitted per data transfer. After each byte, the slave-receiver generates an acknowledge bit (ACK). This is shown in Figure 11-10. When a slave-receiver doesn't acknowledge the slave address or received data, the master must abort the transfer. The slave must leave SDA high so that the master can generate the STOP condition (see Figure 11-7).

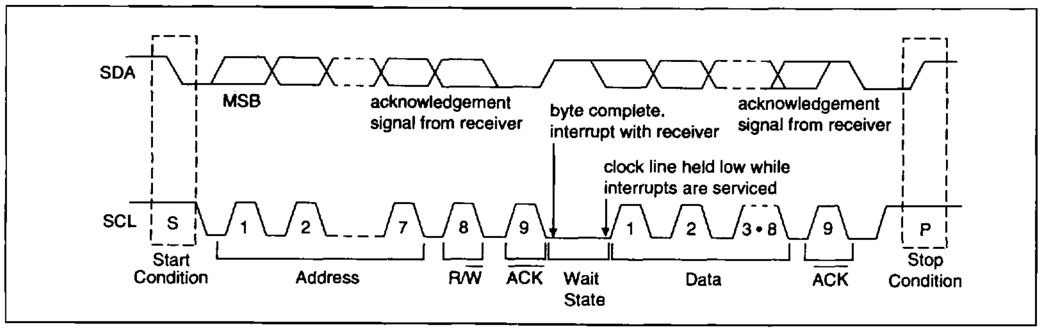
FIGURE 11-10: SLAVE-RECEIVER ACKNOWLEDGE



If the master is receiving the data (master-receiver), it generates an acknowledge signal for each received byte of data, except for the last byte. To signal the end of data to the slave-transmitter, the master does not generate an acknowledge (not acknowledge). The slave then releases the SDA line so the master can generate the STOP condition. The master can also generate the STOP condition during the acknowledge pulse for valid termination of data transfer.

If the slave needs to delay the transmission of the next byte, holding the SCL line low will force the master into a wait state. Data transfer continues when the slave releases the SCL line. This allows the slave to move the received data or fetch the data it needs to transfer before allowing the clock to start. This wait state technique can also be implemented at the bit level. Figure 11-11 shows a data transfer waveform.

FIGURE 11-11: A DATA TRANSFER


SDA
MSB
acknowledgement signal from receiver
byte complete. interrupt with receiver
acknowledgement signal from receiver
clock line held low while interrupts are serviced
SCL
S
1
2
7
8
9
1
2
3•8
9
P
Start Condition
Address
R/W
ACK
Wait State
Data
ACK
Stop Condition

Figures 11-12 and 11-13 show Master-transmitter and Master-receiver data transfer sequences.

When a master does not wish to relinquish the bus (by generating a STOP condition), a repeated START condition (Sr) must be generated. This condition is identical to the start condition (SDA goes high-to-low while SCL is high), but occurs after a data transfer acknowledge pulse (not the bus-free state). This allows a master to send "commands" to the slave and then receive the requested information or to address a different slave device. This sequence is shown in Figure 11-14.

FIGURE 11-12: MASTER-TRANSMITTER SEQUENCE

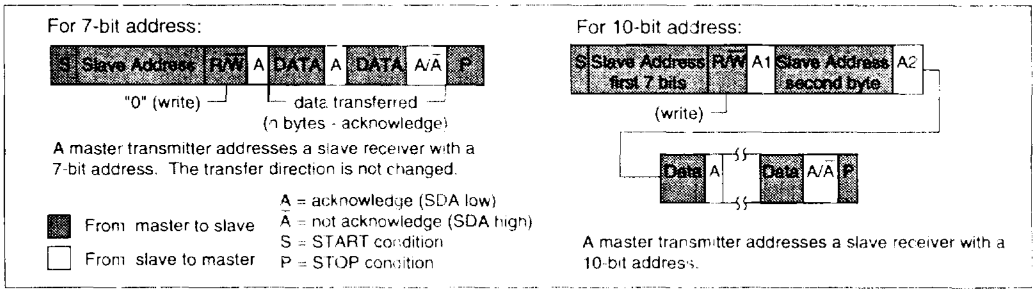

For 7-bit address:
S
Slave Address
R/W
A
DATA
A
DATA
A/Ā
P
"0" (write)
data transferred
(n bytes - acknowledge)
A master transmitter addresses a slave receiver with a 7-bit address. The transfer direction is not changed.
A = acknowledge (SDA low)
Ā = not acknowledge (SDA high)
S = START condition
P = STOP condition
From master to slave
From slave to master
For 10-bit address:
S
Slave Address first 7 bits
R/W
A1
Slave Address second byte
A2
(write)
Data
A
Data
A/Ā
P
A master transmitter addresses a slave receiver with a 10-bit address.

FIGURE 11-13: MASTER-RECEIVER SEQUENCE

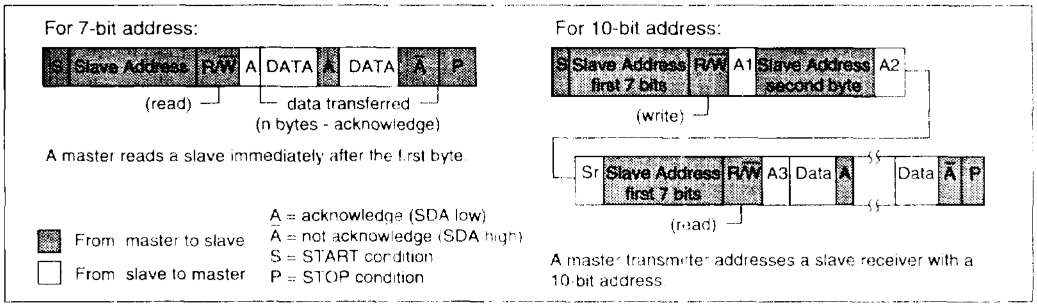
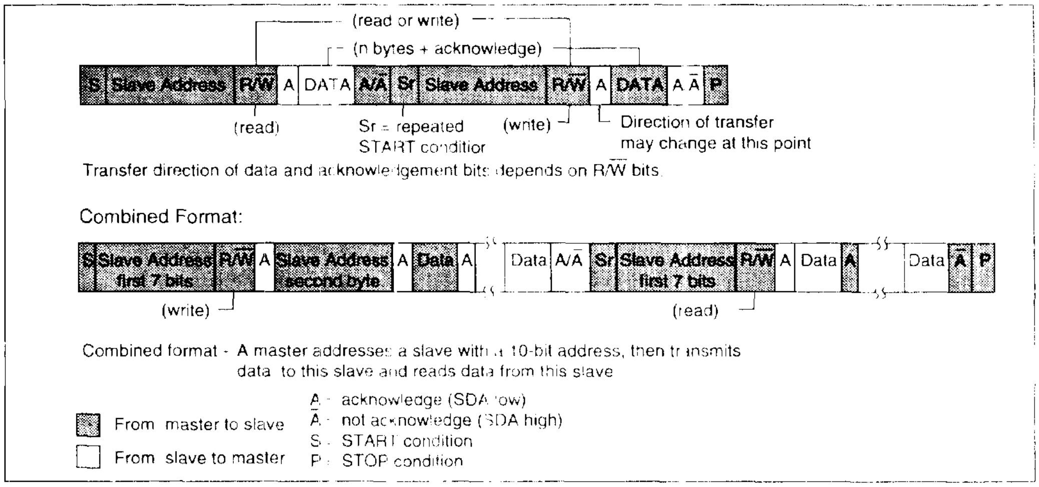

For 7-bit address:
S
Slave Address
R/W
A
DATA
A
DATA
Ā
P
(read)
data transferred
(n bytes - acknowledge)
A master reads a slave immediately after the first byte.
A = acknowledge (SDA low)
Ā = not acknowledge (SDA high)
S = START condition
P = STOP condition
From master to slave
From slave to master
For 10-bit address:
S
Slave Address first 7 bits
R/W
A1
Slave Address second byte
A2
(write)
Sr
Slave Address first 7 bits
R/W
A3
Data
A
Data
Ā
P
(read)
A master transmitter addresses a slave receiver with a 10-bit address.

FIGURE 11-14: COMBINED FORMAT


(read or write)
(n bytes + acknowledge)
S
Slave Address
R/W
A
DATA
A/Ā
Sr
Slave Address
R/W
A
DATA
A
Ā
P
(read)
Sr = repeated START condition
(write)
Direction of transfer may change at this point
Transfer direction of data and acknowledgement bits depends on R/W bits.
Combined Format:
S
Slave Address first 7 bits
R/W
A
Slave Address second byte
A
Data
A
Data
A/Ā
Sr
Slave Address first 7 bits
R/W
A
Data
A
Data
Ā
P
(write)
(read)
Combined format - A master addresses a slave with a 10-bit address, then transmits data to this slave and reads data from this slave.
A = acknowledge (SDA low)
Ā = not acknowledge (SDA high)
S = START condition
P = STOP condition
From master to slave
From slave to master

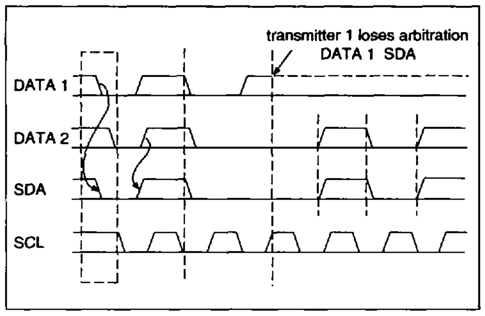
11.2.4 Multi-master

The I²C protocol allows a system to have more than one master. This is called multi-master. When two or more masters try to transfer data at the same time, arbitration and synchronization occur.

11.2.4.1 ARBITRATION

Arbitration takes place on the SDA line, while the SCL line is high. The master which transmits a high when the other master transmits a low loses arbitration (see Figure 11-15), and turns off its data output stage. A master which lost arbitration can generate clock pulses until the end of the data byte where it lost arbitration. When the master devices are addressing the same device, arbitration continues into the data.

FIGURE 11-15: MULTI-MASTER ARBITRATION (2 MASTERS)



Masters that also incorporate the slave function, and have lost arbitration must immediately switch over to slave-receiver mode. This is because the winning master-transmitter may be addressing it.

Arbitration is not allowed between:

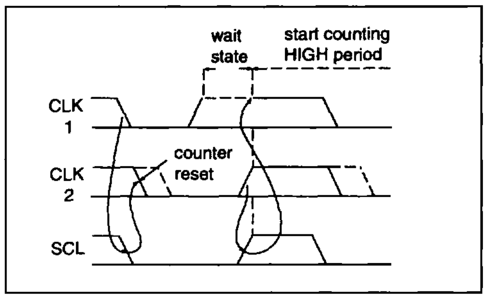
- A repeated START condition
- A STOP condition and a data bit
- A repeated START condition and a STOP condition

Care needs to be taken to ensure that these conditions do not occur.

11.2.4.2 CLOCK SYNCHRONIZATION

Clock synchronization occurs after the devices have started arbitration. This is performed using a wired-AND connection to the SCL line. A high to low transition on the SCL line causes the concerned devices to start counting off their low period. Once a device clock has gone low, it will hold the SCL line low until its SCL high state is reached. The low to high transition of this clock may not change the state of the SCL line, if another device clock is still within its low period. The SCL line is held low by the device with the longest low period. Devices with shorter low periods enter a high wait-state, until the SCL line comes high. When the SCL line comes high, all devices start counting off their high periods. The first device to complete its high period will pull the SCL line low. The SCL line high time is determined by the device with the shortest high period. This is shown in Figure 11-16.

FIGURE 11-16: CLOCK SYNCHRONIZATION

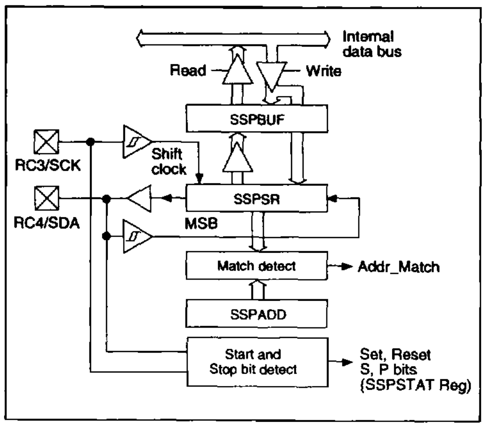


2

11.3 SSP I²C Operation

The SSP module in I²C mode fully implements all slave functions, and provides support in hardware to facilitate software implementations of the master functions. The SSP module implements the standard and fast mode specifications as well as 7-bit and 10-bit addressing. Two pins are used for data transfer. These are the RC3/SCK/SCL pin, which is the clock (SCL), and the RC4/SDI/SDA pin, which is the data (SDA). The user must configure these pins as inputs or outputs through the TRISC<4:3> bits. A block diagram of the SSP module in I²C mode is shown in Figure 11-17. The SSP module functions are enabled by setting the SSP Enable (SSPEN) bit (SSPCON<5>).

FIGURE 11-17: SSP BLOCK DIAGRAM (I²C MODE)



The SSP module has five registers for I²C operation. These are the:

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

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

- I²C Slave mode (7-bit address)
- I²C Slave mode (10-bit address)
- I²C Slave mode (7-bit address), with master-mode support
- I²C Slave mode (10-bit address), with master-mode support
- I²C Master mode, slave is idle

Selection of any I²C mode and with the SSPEN bit set, forces the SCL and SDA pins to be open collector, provided these pins are set to inputs through the TRISC bits.

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

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

The SSPADD register holds the slave address. In 10-bit mode, the user needs to write the high byte of the address (1 1 1 1 0 A9 A8 0). Following the high byte address match, the low byte of the address needs to be loaded (A7 - A0).

11.3.1 Slave Mode

In slave mode, the SCL and SDA pins must be configured as inputs (TRISC<4:3> set). The SSP module will override the input state with the output data when required (slave-transmitter).

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

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

- The Buffer Full (BF) bit was set before the transfer was received.
- The Overflow (SSPOV) bit was set before the transfer was received.

In this case, the SSPSR value is not loaded into the SSPBUF, but the SSPIF bit is set. Table 11-3 shows what happens when a data transfer byte is received, given the status of the BF and SSPOV bits. The shaded boxes shows the condition where user software did not properly clear the overflow condition. The BF flag is cleared by reading the SSPBUF register while the SSPOV bit is cleared through software.

The SCL clock input must have a minimum high and low for proper operation. The high and low times of the I^2C specification as well as the requirement of the SSP module is shown in timing specifications #90 and 91.

11.3.1.1 ADDRESSING

Once the SSP module has been enabled, The SSP waits for a START condition to occur. Following the START condition, the 8-bits are shifted into the SSPSR. All incoming bits are sampled with the rising edge of the clock (SCL) line. The SSPSR<7:1> is compared to the SSPADD register. The address is compared on the falling edge of the eighth clock (SCL) pulse. If the addresses match, and the BF and SSPOV bits are clear, the following things happen:

- SSPSR loaded into SSPBUF
- Buffer Full (BF) bit is set
- $\overline{ACK}$ pulse is generated
- SSP Interrupt Flag (SSPIF) is set (interrupt is generated if enabled) - on falling edge of ninth SCL pulse

In 10-bit address mode, two address bytes need to be received by the slave (see Figure 11-9). The five Most Significant bits (MSbs) of the first address byte, specify if this is a 10-bit address. The R/$\overline{W}$ bit (bit 0) must specify a write, so the slave device will receive the second address byte. For a 10-bit address the first byte would equal '1 1 1 1 0 A9 A8 0', where A9 and A8 are the two MSbs of the address. The sequence of events for 10-bit address are as follows, with steps 7- 9 for slave-transmitter:

1. Receive first (high) byte of Address (SSPIF, BF and UA are set)
2. Update SSPADD with second (low) byte of Address (clears UA and releases SCL line)
3. Read SSPBUF (clears BF) and clear SSPIF
4. Receive second (low) byte of Address (SSPIF, BF and UA are set)
5. Update SSPADD with first (high) byte of Address (clears UA, if match releases SCL line)
6. Read SSPBUF (clears BF) and clear SSPIF
7. Receive Repeated START condition
8. Receive first (high) byte of Address (SSPIF and BF are set)
9. Read SSPBUF (clears BF) and clear SSPIF

TABLE 11-3: DATA TRANSFER RECEIVED BYTE ACTIONS

Status Bits as Data Transfer is Received				
BF	SSPOV	SSPSR → SPBUF	Generate $\overline{ACK}$ Pulse	Set SSPIF bit (SSP Interrupt if Enabled)
0	0	Yes	Yes	Yes
1	0	No	No	Yes
1	1	No	No	Yes
0	1	No	No	Yes

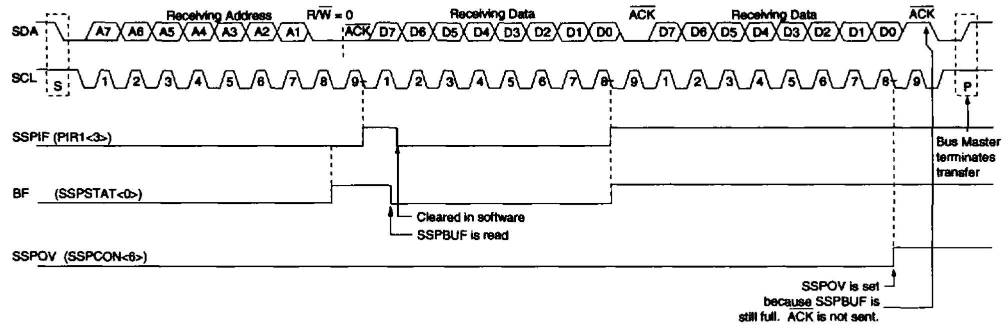
11.3.1.2 RECEPTION

When the $R/\overline{W}$ bit of the address byte is clear and an address match occurs, the $R/\overline{W}$ bit of the SSPSTAT register is cleared. The received address is loaded into the SSPBUF.

When the address byte overflow condition exists then no acknowledge ($\overline{ACK}$) pulse is given. An overflow condition is defined as either the BF bit (SSPSTAT<0>) is set or the SSPOV bit (SSPCON<6>) is set.

A SSPIF interrupt is generated for each data transfer byte. The SSPIF bit must be cleared in software, and the SSPSTAT register is used to determine the status of the byte.

FIGURE 11-18: I²C WAVEFORMS FOR RECEPTION (7-BIT ADDRESS)



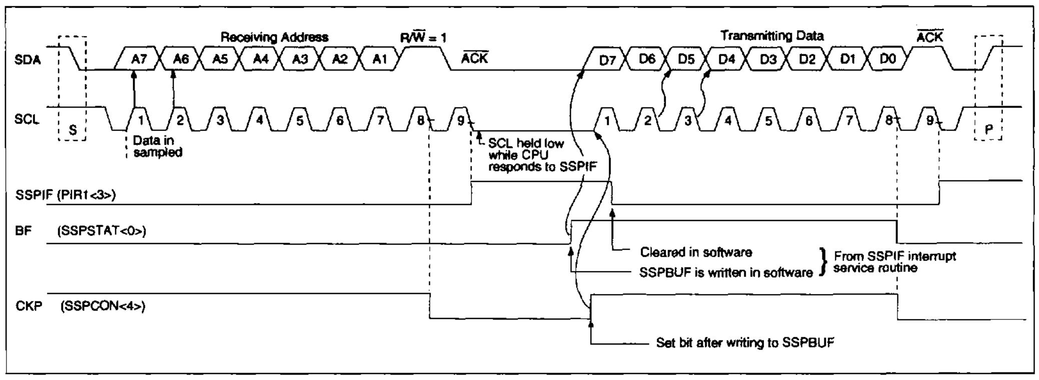
11.3.1.3 TRANSMISSION

When the R/$\overline{W}$ bit of the address byte is set and an address match occurs, the R/$\overline{W}$ bit of the SSPSTAT register is set. The received address is loaded into the SSPBUF. The $\overline{ACK}$ pulse will be sent on the ninth bit, and the SCL pin is held low. The transmit data must be loaded into the SSPBUF register, which also loads the SSPSR register. Then the SCL pin should be enabled by setting the CKP bit (SSPCON<4>). The eight data bits are shifted out on the falling edge of the SCL input. This ensures that the SDA signal is valid during the SCL high time (see Figure 11-19).

A SSPIF interrupt is generated for each data transfer byte. The SSPIF bit must be cleared in software, and the SSPSTAT register is used to determine the status of the byte. The SSPIF bit is set on the falling edge of the ninth clock pulse.

As a slave-transmitter, the $\overline{ACK}$ pulse from the master-receiver is latched on the rising edge of the ninth SCL input pulse. If the SDA line was high (not $\overline{ACK}$), then the data transfer is complete. The slave then monitors for another occurrence of the START bit. If the SDA line was low ($\overline{ACK}$), the transmit data must be loaded into the SSPBUF register, which also loads the SSPSR register. Then the SCL pin should be enabled by setting the CKP bit (SSPCON<4>).

FIGURE 11-19: I²C WAVEFORMS FOR TRANSMISSION (7-BIT ADDRESS)



11.3.2 Master Mode

Master mode of operation is supported by interrupt generation on the detection of the START and STOP conditions. The STOP (P)and START (S) bits are cleared from a reset or when the SSP module is disabled. Control of the I²C bus may be taken when the P-bit is set, or the bus is idle and both the S- and P-bits are cleared.

In master mode the SCL and SDA lines are manipulated by changing the corresponding TRISC<4:3> bit(s) to an output (cleared). The output level is always low, irrespective of the value(s) in PORTC<4:3>. So when transmitting data, a "1" data bit must have the TRISC<4> bit set (input) and a "0" data bit must have the TRISC<4> bit cleared (output). The same scenario is true for the SCL line with the TRISC<3> bit.

The following events will cause the SSP Interrupt Flag (SSPIF) to be set (SSP Interrupt if enabled):

- START condition
- STOP condition
- Data transfer byte transmitted/received

Master mode of operation can be done with either the slave mode idle (SSPM3 - SSPM0 = 1011) or with the slave active. When both master and slave modes are enabled, the software needs to differentiate the source(s) of the interrupt.

11.3.3 Multi-master Mode

In multi-master mode, the interrupt generation on the detection of the START and STOP conditions allows the determination of when the bus is free. The STOP (P)and START (S) bits are cleared from a reset or when the SSP module is disabled. Control of the I²C bus may be taken when the P bit is set, or the bus is idle and both the S and P bits are cleared. When the bus is busy, enabling the SSP Interrupt will generate the interrupt when the stop condition occurs.

In multi-master operation, the SDA line must be monitored to see if the signal level is the expected output level. This check only needs to be done when a high level is output. If a high level is expected and a low level is present, the device needs to release the SDA and SCL lines (set TRISC<4:3>). There are two stages where this arbitration can be lost, these are:

- Address Transfer
- Data Transfer

When the slave logic is enabled, the slave continues to receive. If arbitration was lost during the address transfer stage, the device may be being addressed. If addressed an $\overline{ACK}$ pulse will be generated. If arbitration was lost during the data transfer stage, the device will need to re-transfer the data at a later time.

TABLE 11-4: REGISTERS ASSOCIATED WITH I²C OPERATION

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0B/8B	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
0C	PIR1	PSPIF	—	—	—	SSPIF	CCP1IF	TMR2IF	TMR1IF
8C	PIE1	PSPIE	—	—	—	SSPIE	CCP1IE	TMR2IE	TMR1IE
13	SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register							
93	SSPADD	Synchronous Serial Port (I²C mode) Address Register							
14	SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
94	SSPSTAT	—	—	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF

Legend — = Unimplemented locations, Read as '0'
Note: Shaded boxes are not used by the SSP module in I²C mode.

FIGURE 11-20: OPERATION OF THE I²C MODULE IN IDLE_MODE, RCV_MODE OR XMIT_MODE

```
IDLE_MODE (7-bit):
if (Addr_match)            {   Set interrupt;
                               if (R/$\overline{W}$ = 1)  {Send $\overline{ACK}$ = 0;
                                             set XMIT_MODE;
                                             }
                               else if (R/$\overline{W}$ = 0) set RCV_MODE;
                           }
```

```
RCV_MODE:
if ((SSPBUF=Full) OR (SSPOV = 1))
        {Set SSPOV;
        Do not acknowledge;
        }
else    { transfer SSPSR → SSPBUF;
        send $\overline{ACK}$ = 0;
        }
Receive 8-bits in SSPSR;
Set interrupt;
```

```
XMIT_MODE:
While ((SSPBUF = Empty) AND (CKP=0)) Hold SCL Low;
Send byte;
Set interrupt;
if ($\overline{ACK}$ Received = 1)       {End of transmission;
                                 Go back to IDLE_MODE;
                                 }
else if ($\overline{ACK}$ Received = 0)  Go back to XMIT_MODE;
```

```
IDLE_MODE (10-Bit):
If (High_byte_addr_match AND (R/$\overline{W}$ = 0))
     { PRIOR_ADDR_MATCH = FALSE;
       Set interrupt;
       if ((SSPBUF = Full) OR ((SSPOV = 1))
            { Set SSPOV;
              Do not acknowledge;
            }
       else { Set UA = 1;
              Send $\overline{ACK}$ = 0;
              While (SSPADD not updated) Hold SCL low;
              Clear UA = 0;
              Receive Low_addr_byte;
              Set interrupt;
              Set UA = 1;
              If (Low_byte_addr_match)
                   { PRIOR_ADDR_MATCH = TRUE;
                     Send $\overline{ACK}$ = 0;
                     while (SSPADD not updated) Hold SCL low;
                     Clear UA = 0;
                     Set RCV_MODE;
                   }
            }
     }
else if (High_byte_addr_match AND (R/$\overline{W}$ = 1)
            { if (PRIOR_ADDR_MATCH)
                   { send $\overline{ACK}$ = 0;
                     set XMIT_MODE;
                   }
              else PRIOR_ADDR_MATCH = FALSE;
            }
```

12.0 SPECIAL FEATURES OF THE CPU

What sets apart a microcontroller from other processors are special circuits to deal with the needs of real time applications. The PIC16CXX family has a host of such features intended to maximize system reliability, minimize cost through elimination of external components, provide power saving operating modes and offer code protection. These are:

1. OSC selection
2. Reset
   - Power-On Reset (POR)
   - Power-Up Timer (PWRT)
   - Oscillator Start-Up Timer (OST)
3. Interrupts
4. Watchdog Timer (WDT)
5. SLEEP
6. Code protection
7. In-circuit serial programming

The PIC16CXX has a Watchdog Timer which can be shut off only through EPROM fuses. It runs off its own RC oscillator for added reliability. 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 72ms (nominal) on power-up only, 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.

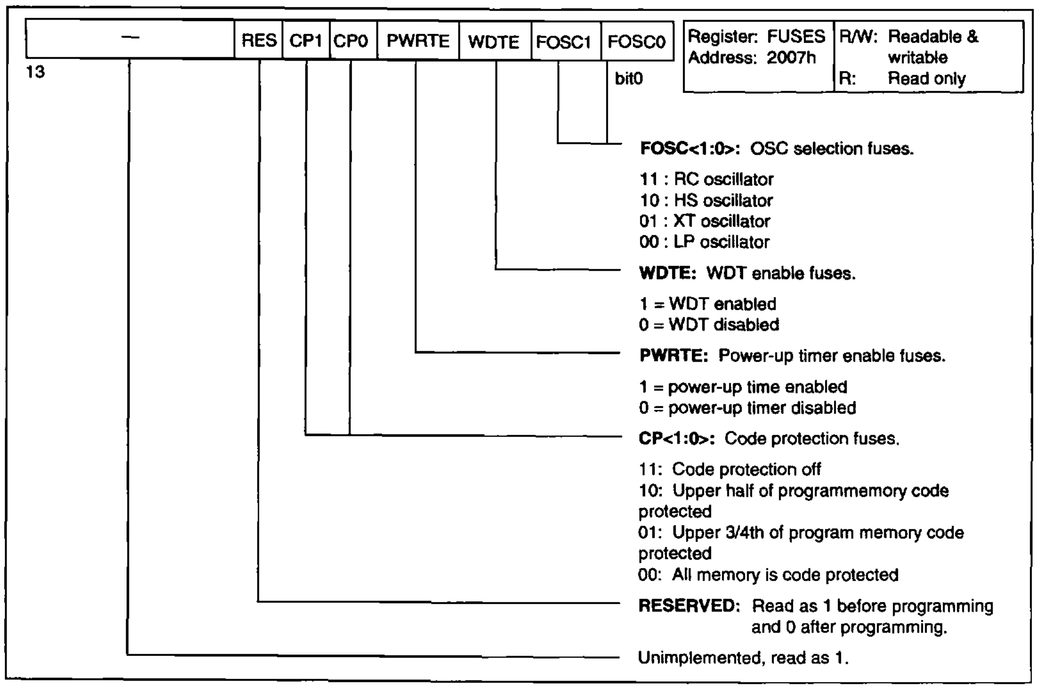
The SLEEP mode is designed to offer a very low current power-down mode. The user can wake-up from SLEEP through external reset, Watchdog Timer timeout or through an interrupt. Several oscillator options are also made available to allow the part to fit the application. The RC oscillator option saves system cost while the LP crystal option saves power. A set of EPROM configuration bits (fuses) are used to select various options.

12.1 Configuration Fuses

The PIC16CXX has six configuration fuses which are EPROM bits. These fuses can be programmed (read as '0') or left unprogrammed (read as '1') to select various device configurations. These bits are mapped in program memory location 2007h.

The user will note that address 2007h is beyond the user program memory space. In fact, it belongs to the special test/configuration memory space (2000h - 3FFFh), which can be accessed during programming.

FIGURE 12-1: CONFIGURATION WORD



Register: FUSES Address: 2007h	R/W: Readable & writable R: Read only

FOSC<1:0>: OSC selection fuses.
11 : RC oscillator
10 : HS oscillator
01 : XT oscillator
00 : LP oscillator

WDTE: WDT enable fuses.
1 = WDT enabled
0 = WDT disabled

PWRTE: Power-up timer enable fuses.
1 = power-up time enabled
0 = power-up timer disabled

CP<1:0>: Code protection fuses.
11: Code protection off
10: Upper half of programmemory code protected
01: Upper 3/4th of program memory code protected
00: All memory is code protected

RESERVED: Read as 1 before programming and 0 after programming.

Unimplemented, read as 1.

12.2 Oscillator Configurations

12.2.1 OSCILLATOR TYPES

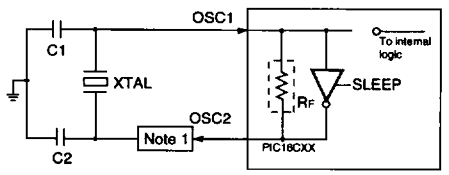
The PIC16CXX can be operated in four different oscillator options. The user can program two configuration bits (FOSC1 and FOSC0) to select one of these four modes:

- LP Low Power Crystal
- XT Crystal
- HS High Speed Crystal
- RC Resistor/Capacitor

12.2.2 CRYSTAL OSCILLATOR / CERAMIC RESONATORS

In XT, LP or HS modes a crystal or ceramic resonator is connected to the OSC1 and OSC2 pins to establish oscillation (Figure 12-2). The PIC16CXX 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. When in XT, LP or HS modes, the device can have an external clock source to drive the OSC1 pin. This is shown in Figure 12-3.

FIGURE 12-2: CRYSTAL OPERATION (OR CERAMIC RESONATOR) (HS, XT OR LP OSC CONFIGURATION)



See Tables 12-1 and 12-2 for recommended values of C1 and C2.

Note1: A series resistor may be required for AT strip cut crystals.

FIGURE 12-3: EXTERNAL CLOCK INPUT OPERATION (HS, XT, or LP OSC CONFIGURATION)

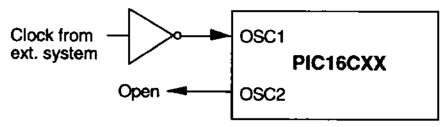


TABLE 12-1: CAPACITOR SELECTION FOR CERAMIC RESONATORS

Oscillator Type	Resonator Frequency	Capacitor Range C1 = C2
XT	455 KHz	150 - 330 pF
	2.0 MHz	20 - 330 pF
	4.0 MHz	20 - 330 pF
HS	8.0 MHz	20 - 200 pF

Higher capacitance increases the stability of oscillator but also increases the start-up time. These values are for design guidance only. Since each resonator has its own characteristics, the user should consult the resonator manufacturer for appropriate values of external components.

TABLE 12-2: CAPACITOR SELECTION FOR CRYSTAL OSCILLATOR

Osc Type	Freq	C1	C2
LP	32 KHz§	15 pF	15 pF
	100 KHz	15 pF	15 pF
	200 KHz	0 - 15 pF	0 - 15 pF
XT	100 KHz	15 - 30 pF	200 - 300 pF
	200 KHz	15 - 30 pF	100 - 200 pF
	455 KHz	15 - 30 pF	15 - 100 pF
	1 MHz	15 - 30 pF	15 - 30 pF
	2 MHz	15 pF	15 pF
	4 MHz	15 pF	15 pF
HS	4 MHz	15 pF	15 pF
	8 MHz	15 pF	15 pF
	20 MHz	15 pF	15 pF

Higher capacitance increases the stability of oscillator but also increases the start-up time. These values are for design guidance only. Rs may be required in HS mode as well as XT mode to avoid overdriving crystals with low drive level specification. Since each crystal has its own characteristics, the user should consult the crystal manufacturer for appropriate values of external components.

§ For $V_{DD} > 4.5V$, C1 = C2 ≃ 30pf is recommended.

12.2.3 EXTERNAL CRYSTAL OSCILLATOR CIRCUIT

Either a prepackaged oscillator can be used or a simple oscillator circuit with TTL gates can be built. Prepackaged oscillators provide a wide operating range and better stability. A well-designed crystal oscillator will provide good performance with TTL gates. Two types of crystal oscillator circuits can be used; one with series resonance, or one with parallel resonance.

Figure 12-4 shows implementation of a parallel resonant oscillator circuit. The circuit is designed to use the fundamental frequency of the crystal. The 74AS04 inverter performs the 180-degree phase shift that a parallel oscillator requires. The 4.7 kΩ resistor provides the negative feedback for stability. The 10 kΩ potentiometer biases the 74AS04 in the linear region. This could be used for external oscillator designs.

FIGURE 12-4: EXTERNAL PARALLEL RESONANT CRYSTAL OSCILLATOR CIRCUIT

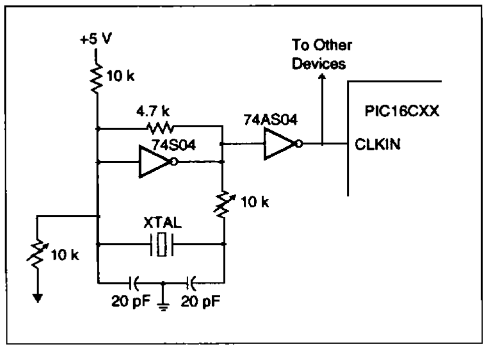
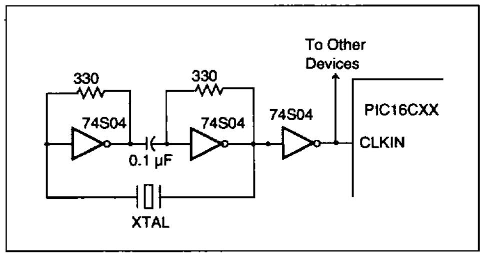


Figure 12-5 shows a series resonant oscillator circuit. This circuit is also designed to use the fundamental frequency of the crystal. The inverter performs a 180-degree phase shift in a series resonant oscillator circuit. The 330-Ω resistors provide the negative feedback to bias the inverters in their linear region.

FIGURE 12-5 : EXTERNAL SERIES RESONANT CRYSTAL OSCILLATOR CIRCUIT



12.2.4 RC OSCILLATOR

For timing insensitive applications the "RC" device option offers additional cost savings. The RC oscillator frequency is a function of the supply voltage, the resistor (Rext) and capacitor (Cext) values, and the operation temperature. In addition to this, the oscillator frequency will vary from unit to unit due to normal process parameter variation. Furthermore, the difference in lead frame capacitance between package types will also affect the oscillation frequency, especially for low Cext values. The user also needs to take into account variation due to tolerance of external R and C components used. Figure 12-6 shows how the R/C combination is connected to the PIC16CXX. For Rext values below 2.2 kOhm, the oscillator operation may become unstable, or stop completely. For very high Rext values (e.g. 1 MOhm), the oscillator becomes sensitive to noise, humidity and leakage. Thus, we recommend to keep Rext between 3 kOhm and 100 kOhm.

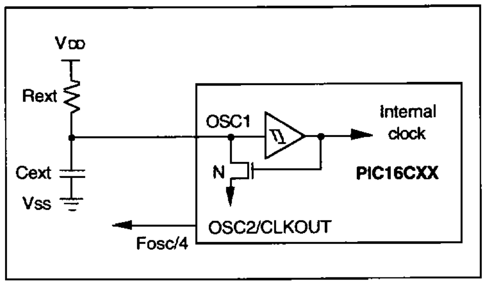
Although the oscillator will operate with no external capacitor (Cext = 0 pF), we recommend using values above 20 pF for noise and stability reasons. With no or small external capacitance, the oscillation frequency can vary dramatically due to changes in external capacitances, such as PCB trace capacitance or package lead frame capacitance.

See Section 17.0 for RC frequency variation from part to part due to normal process variation. The variation is larger for larger R (since leakage current variation will affect RC frequency more for large R) and for smaller C (since variation of input capacitance will affect RC frequency more).

See Section 17.0 for variation of oscillator frequency due to V_{DD} for given Rext/Cext values as well as frequency variation due to operating temperature for given R, C, and V_{DD} values.

The oscillator frequency, divided by 4, is available on the OSC2/CLKOUT pin, and can be used for test purposes or to synchronize other logic (see Figure 3-2 for waveform).

FIGURE 12-6: RC OSCILLATOR MODE



12.3 Reset

The PIC16CXX differentiates between various kinds of reset:

a) Power-On Reset (POR)
b) MCLR Reset during normal operation
c) MCLR reset during SLEEP
d) WDT time-out reset during normal operation
e) WDT time-out reset during SLEEP

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 (POR), on MCLR or WDT reset during normal operation and on MCLR reset during SLEEP. They are not affected by a WDT reset during SLEEP, since this reset is viewed as the resumption of normal operation. TO and PD bits are set or cleared differently in different reset situations as indicated in Table 12-4. These bits are used in software to determine the nature of reset. See Table 12-6 for a full description of reset states of all registers.

A simplified block diagram of the on-chip reset circuit is shown in Figure 12-7.

12.4 Power-On Reset (POR), Power-Up-Timer (PWRT) and Oscillator Start-up Timer (OST)

12.4.1 Power-On Reset (POR)

A Power-On Reset pulse is generated on-chip when VDD rise is detected (in the range of 1.6V - 1.8V). To take advantage of the POR, just tie MCLR pin directly (or through a resistor) to VDD. This will eliminate external RC components usually needed to create Power-On Reset. A minimum rise time for VDD is required. See Electrical Specifications for details.

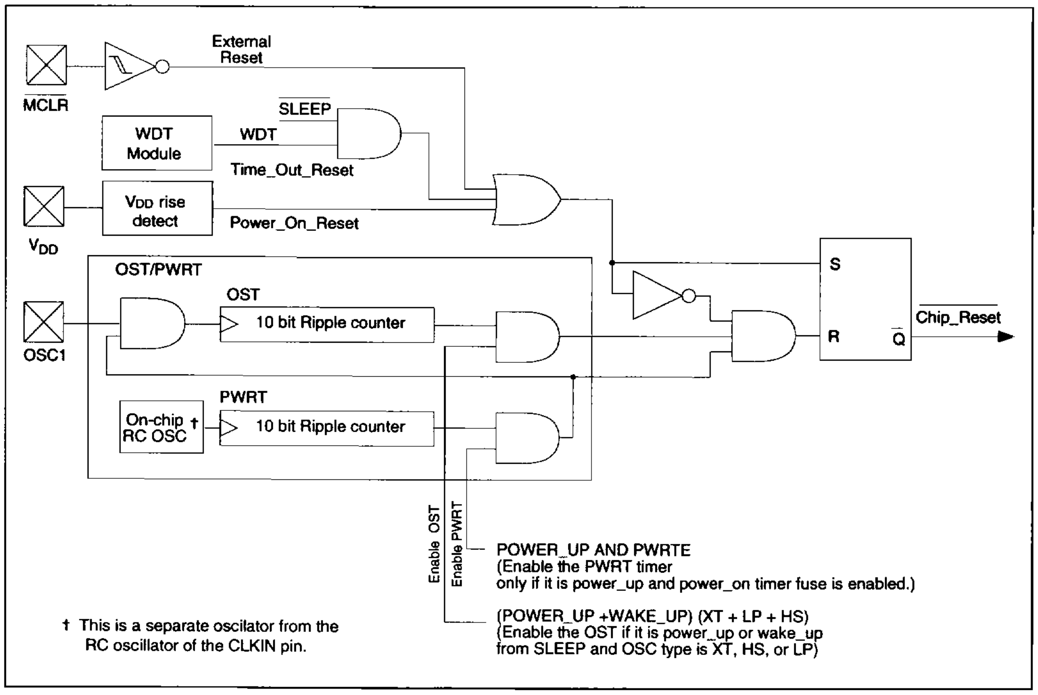
The POR circuit does not produce internal reset when VDD declines.

12.4.2 POWER-UP TIMER (PWRT)

The Power-Up Timer provides a fixed 72ms time-out on power-up only from POR. The power-up timer operates on an internal RC oscillator. The chip is kept in reset as long as PWRT is active. The PWRT delay allows the VDD to rise to an acceptable level. A configuration fuse, PWRTE can enable (if set) or disable (if cleared or programmed) the power-up timer.

The Power-Up Time delay will vary from chip to chip and due to VDD and temperature. See the DC Parameters section for details.

FIGURE 12-7: SIMPLIFIED BLOCK DIAGRAM OF ON-CHIP RESET CIRCUIT


External Reset
MCLR
SLEEP
WDT Module
WDT
Time_Out_Reset
VDD rise detect
Power_On_Reset
VDD
OST/PWRT
OST
10 bit Ripple counter
OSC1
PWRT
On-chip † RC OSC
10 bit Ripple counter
S
R
Q
Chip_Reset
Enable OST
Enable PWRT
POWER_UP AND PWRTE
(Enable the PWRT timer
only if it is power_up and power_on timer fuse is enabled.)
(POWER_UP +WAKE_UP) (XT + LP + HS)
(Enable the OST if it is power_up or wake_up
from SLEEP and OSC type is XT, HS, or LP)
† This is a separate oscillator from the RC oscillator of the CLKIN pin.

12.4.3 OSCILLATOR START-UP TIMER (OST)

The Oscillator Start-Up Timer (OST) provides 1024 oscillator cycle (from OSC1 input) delay after the PWRT delay is over. This guarantees that the crystal oscillator or resonator has started and stabilized.

The OST time-out is invoked only for XT, LP and HS modes and only on Power-On Reset or wake-up from SLEEP.

12.4.4 TIME-OUT SEQUENCE

On power-up the time-out sequence is as follows: First PWRT time-out is invoked after POR has expired. Then OST is activated. The total time-out will vary based on oscillator configuration and PWRTE fuse status. For example, in RC mode with PWRTE fuse cleared (PWRT disabled), there will be no time-out at all. Figures 12-8 to 12-10 depict time-out sequences.

TABLE 12-3: TIME-OUT IN VARIOUS SITUATIONS

Oscillator Configuration	Power up		Wake up from SLEEP
	PWRTE = 1	PWRTE = 0	
XT, HS, LP	72 ms + 1024 tosc	1024 tosc	1024 tosc
RC	72 ms	–	–

Since the time-outs occur from POR pulse, if $\overline{\text{MCLR}}$ is kept low long enough, the time-outs will expire. Then bringing $\overline{\text{MCLR}}$ high will begin execution immediately (see Figure 12-9). This is useful for testing purposes or to synchronize more than one PIC16CXX device operating in parallel.

Table 12-5 shows the reset conditions for some special registers, while Table 12-6 shows the reset conditions for all the registers.

12.4.1 POWER CONTROL/STATUS REGISTER (PCON)

The power control/status register, PCON (address 8Eh) has only one bit.

Bit1 is $\overline{\text{POR}}$ (Power-on-reset). It is cleared on power-on-reset and unaffected otherwise. The user must set this bit following power-on-reset. On a subsequent reset if $\overline{\text{POR}}$ is 'cleared, it will indicate that a Power-On Reset must have occured (VDD may have gone too low).

TABLE 12-4: STATUS BITS AND THEIR SIGNIFICANCE

$\overline{\text{POR}}$	$\overline{\text{TO}}$	$\overline{\text{PD}}$	
0	1	1	Power-on-reset
0	0	X	Illegal, $\overline{\text{TO}}$ is set on POR
0	X	0	Illegal, $\overline{\text{PD}}$ is set on POR
1	0	1	WDT reset during normal operation
1	0	0	WDT timeout wakeup from SLEEP
1	1	1	$\overline{\text{MCLR}}$ reset during normal operation
1	1	0	$\overline{\text{MCLR}}$ reset during SLEEP or interrupt wake-up from SLEEP

TABLE 12-5: RESET CONDITION FOR SPECIAL REGISTERS

	PCL Addr: 02h	STATUS Addr: 03h	PCON Addr: 8Eh
Power-On Reset	000h	0001 1xxx	---- --0-
$\overline{\text{MCLR}}$ reset during normal operation	000h	0001 1uuu	---- --u-
$\overline{\text{MCLR}}$ reset during SLEEP	000h	0001 0uuu	---- --u-
WDT reset during normal operation	000h	0000 1uuu	---- --u-
WDT during SLEEP	PC + 1	uuu0 0uuu	---- --u-
Interrupt wake-up from SLEEP	PC + 1 (1)	uuu1 0uuu	---- --u-

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

Notes: 1. When the wake-up is due to an interrupt and the GIE bit is set, the PC is loaded with the interrupt vector (0004h).

TABLE 12-6: RESET CONDITION FOR REGISTERS

Register	Address	• Power-on Reset	• $\overline{\text{MCLR}}$ Reset during: - normal operation - SLEEP • WDT timeout during normal operation	• Wake up from SLEEP through interrupt • Wake up from SLEEP through WDT timeout
W	-	xxxx xxxx	uuuu uuuu	uuuu uuuu
INDF	00h	-	-	-
TMR0	01h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PC	02h	0000h	0000h	PC + 1 (2)
STATUS	03h	0001 1xxx	000? ?uuu (3)	uuu? ?uuu (3)
FSR	04h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTA	05h	--xx xxxx	--uu uuuu	--uu uuuu
PORTB	06h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTC	07h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTD	08h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTE	09h	---- -xxx	---- -uuu	---- -uuu
PCLATH	0Ah	---0 0000	---0 0000	---u uuuu
INTCON	0Bh	0000 000x	0000 0000	uuuu uuuu(1)
PIR1	0Ch	0--- 0000	0--- 0000	u--- uuuu(1)
TMR1L	0Eh	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR1H	0Fh	xxxx xxxx	uuuu uuuu	uuuu uuuu
T1CON	10h	--00 0000	--uu uuuu	--uu uuuu
TMR2	11h	xxxx xxxx	uuuu uuuu	uuuu uuuu
T2CON	12h	-000 0000	-000 0000	-uuu uuuu
SSPBUF	13h	xxxx xxxx	uuuu uuuu	uuuu uuuu
SSPCON	14h	0000 0000	0000 0000	uuuu uuuu
CCPR1L	15h	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR1H	16h	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCP1CON	17h	--00 0000	--00 0000	--uu uuuu
INDF	80h	-	-	-
OPTION	81h	1111 1111	1111 1111	uuuu uuuu
PC	82h	0000h	0000h	PC + 1
STATUS	83h	0001 1xxx	000? ?uuu (3)	uuu? ?uuu (3)
FSR	84h	xxxx xxxx	uuuu uuuu	uuuu uuuu
TRISA	85h	--11 1111	--11 1111	--uu uuuu
TRISB	86h	1111 1111	1111 1111	uuuu uuuu
TRISC	87h	1111 1111	1111 1111	uuuu uuuu
TRISD	88h	1111 1111	1111 1111	uuuu uuuu
TRISE	89h	0000 -111	0000 -111	uuuu -uuu
PCLATH	8Ah	---0 0000	---0 0000	---u uuuu
INTCON	8Bh	0000 000x	0000 0000	uuuu uuuu(1)
PIE1	8Ch	0--- 0000	0--- 0000	u--- uuuu
PCON	8Eh	---- --0-	---- --1-	---- --1-
PR2	92h	xxxx xxxx	uuuu uuuu	uuuu uuuu
SSPADD	93h	0000 0000	0000 0000	uuuu uuuu
SSPSTAT	94h	--00 0000	--00 0000	--uu uuuu

Legend: u = unchanged, x = unknown, - = unimplemented bit, reads as '0', ? = value depends on condition

Notes: 1. One or more bits in INTCON, PIR1 and/or PIR2 will be affected (to cause wake-up).
2. When the wake-up is due to an interrupt and the GIE bit is set, the PC is loaded with the interrupt vector (0004h).
3. See Table 12-5 for reset value for specific condition.

FIGURE 12-8 : TIME-OUT SEQUENCE ON POWER-UP ($\overline{\text{MCLR}}$ NOT TIED TO VDD): Case 1

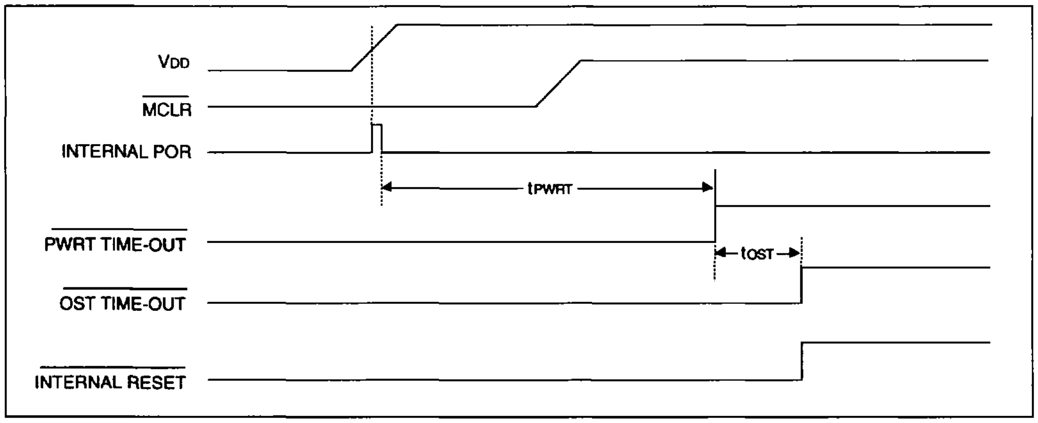


FIGURE 12-9: TIME-OUT SEQUENCE ON POWER-UP ($\overline{\text{MCLR}}$ NOT TIED TO VDD): Case 2

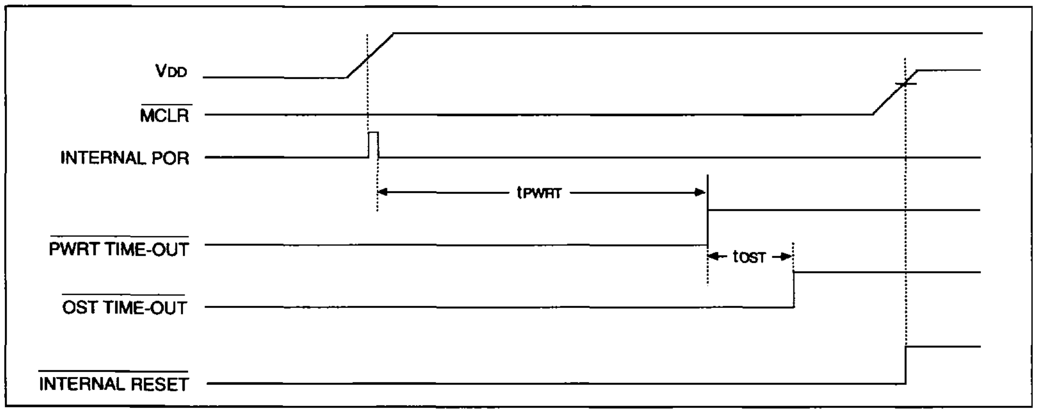


FIGURE 12-10: TIME-OUT SEQUENCE ON POWER-UP ($\overline{\text{MCLR}}$ TIED TO VDD)

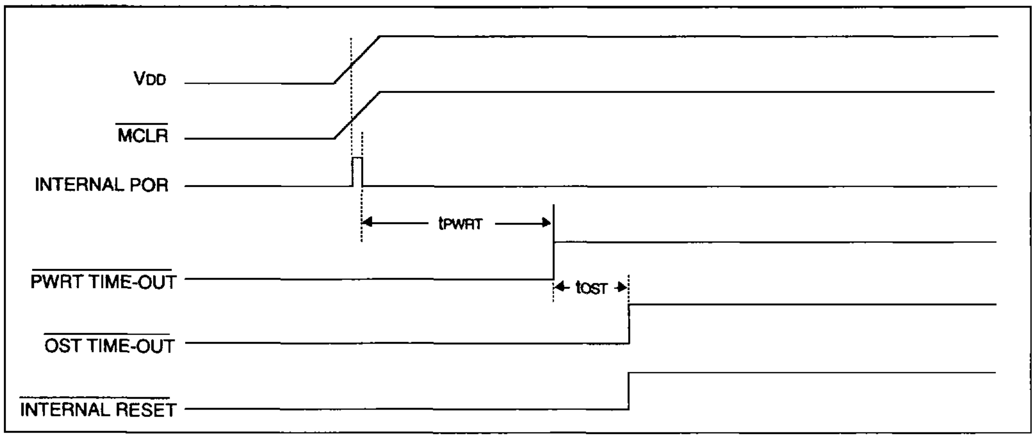
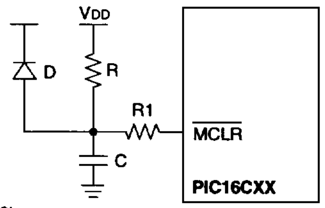


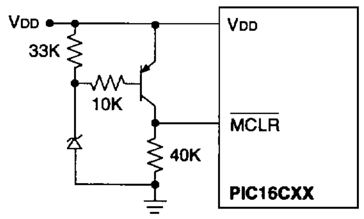
FIGURE 12-11: EXTERNAL POWER-ON RESET CIRCUIT (FOR SLOW VDD POWER-UP)



Notes:

1. External power-on reset circuit is required only if VDD power-up slope is too slow. The diode D helps discharge the capacitor quickly when VDD powers down.
2. R < 40KΩ is recommended to make sure that voltage drop across R does not exceed 0.2V (max leakage current spec on $\overline{\text{MCLR}}$ pin is 5μA). A larger voltage drop will degrade V_{IH} level on $\overline{\text{MCLR}}$ pin.
3. R1 = 100Ω to 1KΩ will limit any current flowing into $\overline{\text{MCLR}}$ from external capacitor C in the event of $\overline{\text{MCLR}}$ pin breakdown due to ESD or EOS.

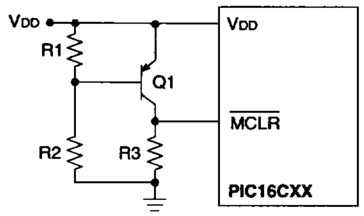
FIGURE 12-12: BROWN-OUT PROTECTION CIRCUIT 1



Notes:

1. This circuit will activate reset when VDD goes below (V_Z + 0.7V) where V_Z = Zener voltage.

FIGURE 12-13: BROWN-OUT PROTECTION CIRCUIT 2



Notes:

1. This brown circuit is less expensive, albeit less accurate. Transistor Q1 turns off when VDD is below a certain level such that:

$$V_{DD} \cdot \frac{R1}{R1 + R2} = 0.7V.$$

12.5 Interrupts

The PIC16C64 has eight sources of interrupt:

- External interrupt RB0/INT
- RTCC timer/counter overflow interrupt
- PortB change interrupts (pins RB<7:4>)
- Timer1 overflow interrupt
- Timer2 interrupt
- CCP1 interrupt
- Sync serial port interrupt
- Microprocessor port read/write interrupt

The interrupt control register (INTCON, addr 0Bh) records individual interrupt requests in flag bits. It also has individual and global interrupt enable bits.

A 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 `RETFIE` instruction allows user to return from interrupt and enable interrupt at the same time.

The INT pin interrupt, the RB port change interrupt and the RTCC overflow interrupt flags are contained in the INTCON register.

The peripheral interrupt flags are contained in the special register PIR1 (0Ch). The corresponding interrupt enable bits are contained in special registers PIE1 (8Ch).

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.

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

Note 2: If an interrupt occurs while the Global Interrupt Enable (GIE) bit is being cleared, the GIE bit may unintentionally be re-enabled by the user's Interrupt Service Routine (the `RETFIE` instruction). The events that would cause this to occur are:

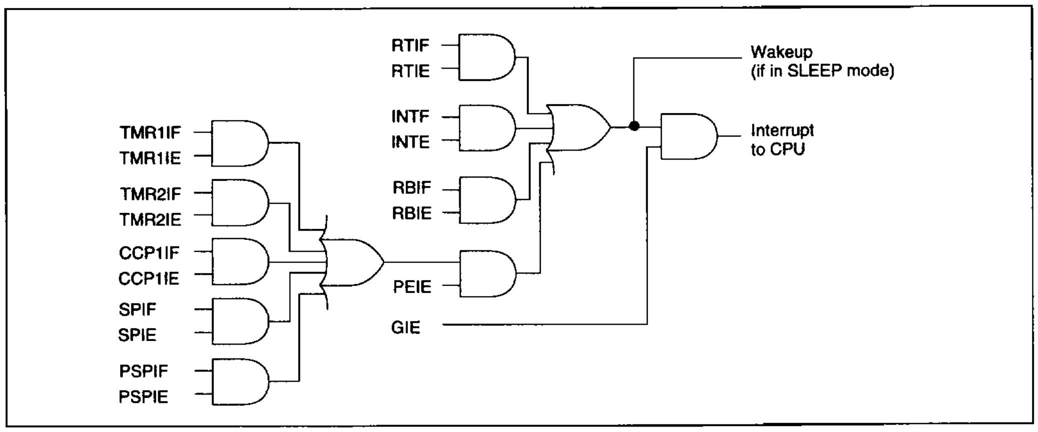
1. An instruction clears the GIE bit while an interrupt is acknowledged.
2. The program branches to the Interrupt vector and executes the Interrupt Service Routine.
3. The Interrupt Service Routine completes with the execution of the `RETFIE` instruction. This causes the GIE bit to be set (enables interrupts), and the program returns to the instruction after the one which was meant to disable interrupts.

The method to ensure that interrupts are globally disabled is:

1. Ensure that the GIE bit was cleared by the instruction, as shown in the following code:

```
LOOP  BCF   INTCON, GIE  ; Disable Global
                         ;   Interrupts
      BTFSC INTCON, GIE  ; Global Interrupts
                         ;   Disabled?
      GOTO  LOOP         ; NO, try again
      :                  ; Yes, continue
                         ;   with program
                         ;   flow
```

FIGURE 12-14: INTERRUPT LOGIC



12.5.1 INT INTERRUPT

External interrupt on RB0/INT pin is edge triggered: either rising if INTEDG bit (OPTION<6>) is set, or falling, if INTEDG bit is clear. When a valid edge appears on INT pin, the INTF bit (INTCON<1>) is set. This interrupt can be disabled by clearing the INTE control bit (INTCON<4>). INTF bit must be cleared in software in the interrupt service routine before re-enabling this interrupt. The INT interrupt can wake up the processor from SLEEP, if the INTE bit was set prior to going into SLEEP. The status of the GIE bit decides whether or not the processor branches to the interrupt vector following wake-up. See Section 12.8 for details on SLEEP and Figure 12-16 for timing of wake-up from SLEEP through INT interrupt.

12.5.2 TMR0 INTERRUPT

An overflow (FFh → 00h) in the TMR0 will set the T0IF (INTCON<2>) bit. The interrupt can be enabled/disabled by setting/clearing T0IE (INTCON<5>) bit. For operation of the TMR0 module, see Section 7.0.

12.5.3 PORT RB INTERRUPT

An input change on PORTB <7:4> will set the RBIF (INTCON<0>) bit. The interrupt can be enabled/disabled by setting/clearing the RBIE (INTCON<4>) bit. For operation of PORTB, see Section 5.2.

12.6 Context Saving During Interrupts

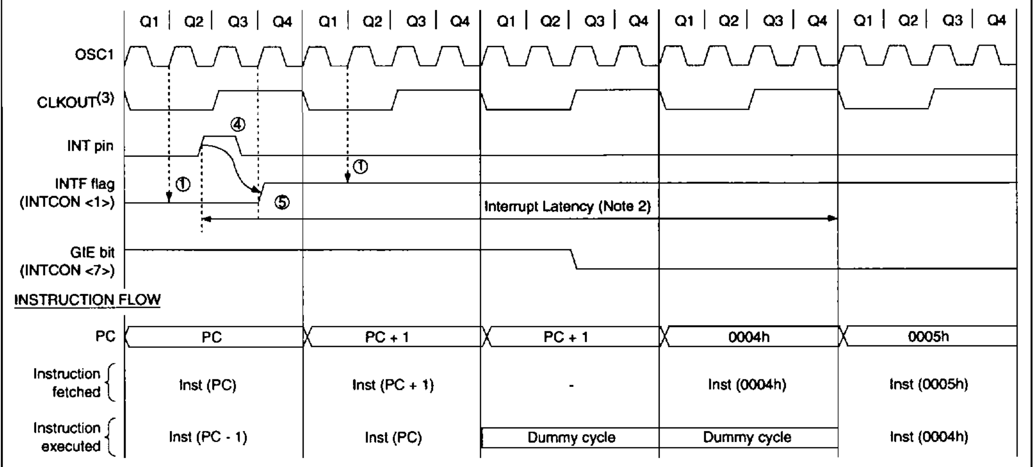
During an interrupt, only the return PC value is saved on the stack. Typically, users may wish to save key registers during an interrupt e.g. W register and status register. This will have to be implemented in software.

EXAMPLE 12-1: SAVING W REGISTER AND STATUS IN RAM:

```
push:  movwf   temp_w          ;Saving Values
       swapf   STATUS,W        ;
       movwf   temp_stat       ;
       :                       ;Interrupt
       :                       ; Service Routine
pop:   swapf   temp_stat,W     ;Restoring
                               ; Values
       movwf   STATUS          ;
       swapf   temp_w, F       ;Do not want to
       swapf   temp_w, W       ; affect the
                               ; z-bit
```

2

FIGURE 12-15: INT PIN INTERRUPT TIMING



Notes:
1. INTF flag is sampled here (every Q1)
2. Interrupt latency = 3-4 Tcy where Tcy = instruction cycle time. Latency is the same whether Inst (PC) is a single cycle or a 2-cycle instruction.
3. CLKOUT is available only in RC oscillator mode.
4. For minimum width spec of INT pulse, refer to AC specs.
5. INTF is enabled to be set anytime during the Q4-Q1 cycles.

12.7 Watchdog Timer (WDT)

The watchdog timer is realized as a free running on-chip RC oscillator which does not require any external components. This RC oscillator is separate from the RC oscillator of the CLKIN pin. That means that the WDT will run, even if the clock on the OSC1 and OSC2 pins of the device has been stopped, for example, by execution of a SLEEP instruction. During normal operation, a WDT timeout generates a device RESET. If the device is in SLEEP mode, a WDT timeout causes the device to wake-up and continue with normal operation. The WDT can be permanently disabled by programming the configuration fuse WDTE as a '0' (Section 12.1).

12.7.1 WDT PERIOD

The WDT has a nominal time-out period of 18 ms, (with no prescaler). 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, a prescaler with a division ratio of up to 1:128 can be assigned to the WDT under software control by writing to the OPTION register. Thus, time-out periods up to 2.3 seconds can be realized.

The CLRWDT and SLEEP instructions clear the WDT and the prescaler, if assigned to the WDT, and prevent it from timing out and generating a device RESET condition.

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

12.7.2 WDT PROGRAMMING CONSIDERATIONS

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

FIGURE 12-16: WATCHDOG TIMER BLOCK DIAGRAM

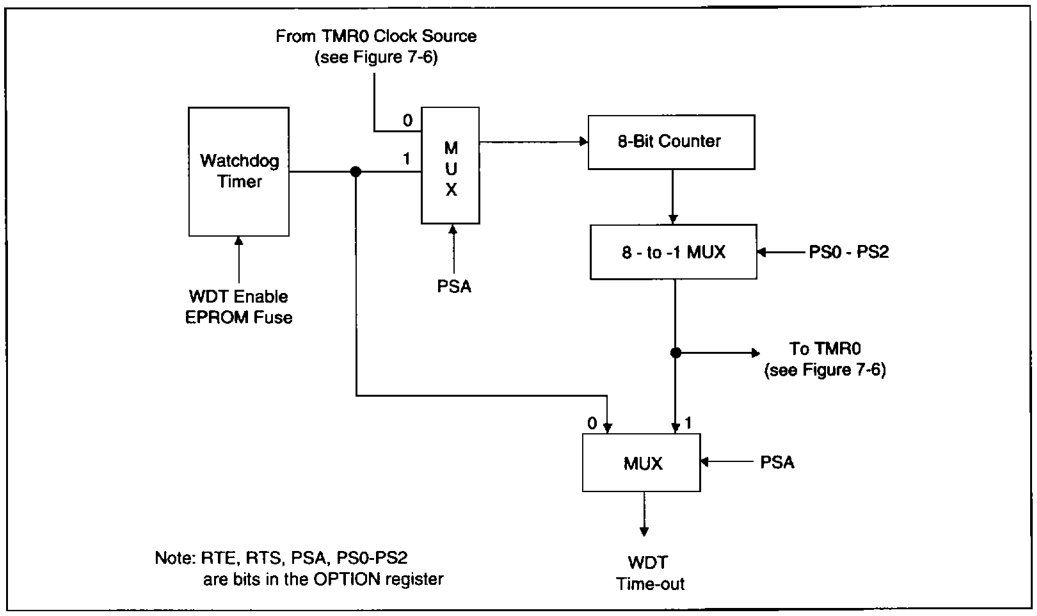


FIGURE 12-17: SUMMARY OF WATCHDOG TIMER REGISTERS

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
2007	Config. Fuses	—	—	CP1	CP0	PWRTE	WDTE	FOSC1	FOSC0
81	OPTION	RBPU	INTEDG	RTS	RTE	PSA	PS2	PS1	PS0

12.8 Power-Down Mode (SLEEP)

The Power-Down mode is entered by executing a `SLEEP` instruction.

If enabled, the Watchdog Timer will be cleared but keeps running, the $\overline{PD}$ bit in the STATUS register is cleared, the $\overline{TO}$ bit is set, and the oscillator driver is turned off. The I/O ports maintain the status they had, before the SLEEP command was executed (driving high, low, or hi-impedance).

For lowest curent consumption in this mode, all I/O pins should be either at VDD, or Vss, with no external circuitry drawing current from the I/O pin. I/O pins that are hi-impedence inputs should be pulled high or low externally to avoid switching currents caused by floating inputs. The RTCC input should also be at VDD or Vss for lowest current consumption. The contribution from on chip pull-ups on PortB should be considered.

The $\overline{MCLR}$ pin must be at a logic high level (VIHMC).

It should be noted that a RESET generated by a WDT time-out does not drive $\overline{MCLR}$ pin low.

14.8.1 WAKE-UP FROM SLEEP

The device can wake up from SLEEP through one of the following events:

1. External reset input on $\overline{MCLR}$ pin
2. Watchdog timer time-out reset (if WDT was enabled)
3. Interrupt from INT pin, RB port change, TMR0 overflow, or some Peripheral Interrupts.

The following peripheral interrupts can wake-up from SLEEP:

1. TMR1 interrupt. Timer1 must be operating as an asynchronous counter.
2. SSP Start/Stop bit detect interrupt.
3. CCP capture mode interrupt.
4. Slave port read or write.

Other peripherals can not generate interrupts since during SLEEP, no on-chip Q clocks are present.

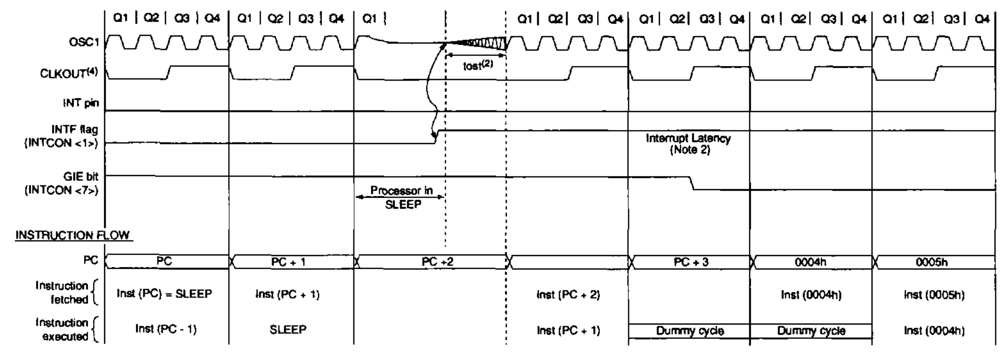
The first event will cause a device reset. The two latter events are considered a continuation of program execution. The $\overline{TO}$ and $\overline{PD}$ bits in the STATUS register can be used to determine the cause of device reset. $\overline{PD}$ bit, which is set on power-up is cleared when SLEEP is invoked. $\overline{TO}$ bit is cleared if WDT time-out occurred (and caused wake-up).

When the `SLEEP` instruction is being executed, the next instruction (PC + 1) is pre-fetched. For the device to wake-up through an interrupt event, the corresponding interrupt enable bit must be set (enabled). Wake-up is regardless of the state of the GIE bit. If the GIE bit is clear (disabled), the device continues execution at the instruction after the `SLEEP` instruction. If the GIE bit is set (enabled), the device executes the instruction after the `SLEEP` instruction and the branches to the interrupt address (0004h). In cases where the execution of the instruction following `SLEEP` is not desirable, the user should have a `NOP` after the `SLEEP` instruction.

Note: **If the global interrupts are disabled (GIE is cleared), but any interrupt source has both its interrupt enable bit and the corresponding interrupt flag bits set, the device will immediately wake from sleep.**

The WDT is cleared when the device wakes-up from sleep, regardless of the source of wake-up.

FIGURE 12-18: WAKE-UP FROM SLEEP THROUGH INTERRUPT



Notes:
1. XT, HS or LP oscillator mode assumed
2. tost = 1024 tosc (drawing not to scale). This delay will not be there for RC osc mode.
3. GIE = 1 assumed. In this case after wake up processor jumps to interrupt routine.
If GIE = 0, execution will continue in line.
4. CLKOUT is not available in these osc modes, but shown here for timing reference.

12.9 Code Protection

The code in the program memory can be protected by blowing the code protect fuses (CP<1:0>).

When code protected, the contents of the program memory cannot be read out in a way that the program code can be reconstructed.

In code-protected mode, the configuration word (2007h) will not be scrambled, allowing reading of all fuses.

12.9.1 CODE PROTECTION FUSES:

The PIC16/17 microcontroller code protection scheme has been enhanced. These enhancements allow the user to selectively code protect portions of the program memory. The two code protect fuses (CP<1:0>) allow the selection of the following code protection mappings:

11	Program memory not code protected
01	Upper 3/4 of program memory code protected (200h - 7FF)
10	Upper 1/2 of program memory code protected (400h - 7FF)
00	Entire program memory code protected

Any word of a protected memory section will read out a scrambled version as shown in Figure 12-17:

FIGURE 12-19: PROTECTED MEMORY READ FORMAT

13	7 6	0
0 0 0 0	0 0 0	b6 b5 b4 b3 b2 b1 b0

where b0 = XNOR of bit0 and bit7 of the program memory location.
b1 = XNOR of bit1 and bit8 of the programmemory location.
•
•
•
b6 = XNOR of bit6 and bit13 of the program.

The configuration word is not code protected, and therefore no scrambling is done. Unprotected segments read normally. Once the program memory location(s) have been code protected, those memory locations can not be further programmed.

12.9.2 VERIFYING A CODE-PROTECTED PART

When code protected, verifying any program memory location will read a scrambled output which looks like "0000000xxxxxxx" (binary) where X is 1 or 0. To verify a device after code protection, follow this procedure:

1. First, program and verify a good device without code protecting it.
2. Next, blow its code protection fuse and then load its contents in a file.
3. Verify any code-protected PIC16C64 against this file.

12.10 In-Circuit Serial Programming

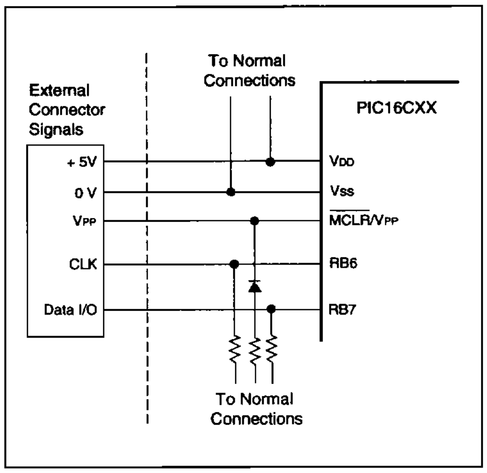
The PIC16CXX microcontrollers can be serially programmed while in the end application circuit. This is simply done with two lines for clock and date and three other lines for power, ground and programming voltage. This allows customers to manufacture boards with unprogrammed devices, and then program the microcontroller just before shipping the product. This 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 $\overline{MCLR}$ (Vpp) pin from Vil to Vihh. RB6 becomes the programming clock and RB7 becomes the programming data. Both RB6 and RB7 are schmidt 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 date 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 #DS30153).

A typical in-system serial programming connection is shown in Figure 12-18.

FIGURE 12-20: TYPICAL IN-SYSTEM SERIAL PROGRAMMING CONNECTION



13.0 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 13-2 lists byte-oriented, bit-oriented, and literal and control operations. Table 13-1 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 utilized 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 8- or 11-bit constant or literal value.

TABLE 13-1: 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 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
→	Assigned to
< >	Register bit field
∈	In the set of
italics	User defined term

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 µsec. If a conditional test is true or the program counter is changed as a result of an instruction, the instruction execution time is 2 µsec.

Table13-2 lists the instructions recognized by the MPASM assembler.

Figure 13-1 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 hexidecimal number:

0xhh

where h signifies a hexidecimal digit.

FIGURE 13-1: GENERAL FORMAT FOR INSTRUCTIONS

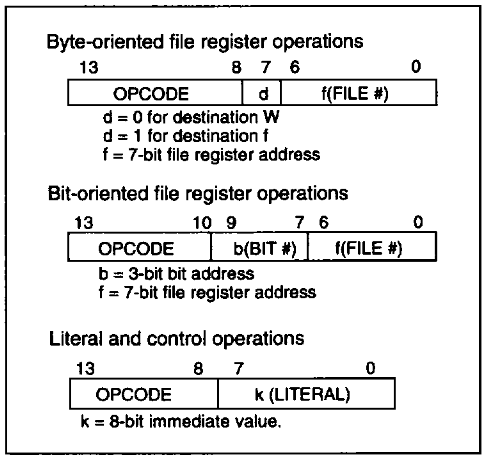

Byte-oriented file register operations
13 8 7 6 0
OPCODE
d
f(FILE #)
d = 0 for destination W
d = 1 for destination f
f = 7-bit file register address
Bit-oriented file register operations
13 10 9 7 6 0
OPCODE
b(BIT #)
f(FILE #)
b = 3-bit bit address
f = 7-bit file register address
Literal and control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

TABLE 13-2: INSTRUCTION SET

Mnemonic, Operands	Description	Cycles	14-Bit Opcode	Status Affected	Notes
			msb lsb		
BYTE-ORIENTED FILE REGISTER OPERATIONS					
ADDWF f, d	Add W and f	1	00 0111 dfff ffff	C, DC, Z	1, 2
ANDWF f, d	AND W and f	1	00 0101 dfff ffff	Z	1, 2
CLRF f	Clear f	1	00 0001 1fff ffff	Z	2
CLRW -	Clear W	1	00 0001 0xxx xxxx	Z	
COMF f, d	Complement f	1	00 1001 dfff ffff	Z	1, 2
DECF f, d	Decrement f	1	00 0011 dfff ffff	Z	1, 2
DECFSZ f, d	Decrement f, Skip if 0	1 (2)	00 1011 dfff ffff		1, 2, 3
INCF f, d	Increment f	1	00 1010 dfff ffff	Z	1, 2
INCFSZ f, d	Increment f, Skip if 0	1 (2)	00 1111 dfff ffff		1, 2, 3
IORWF f, d	Inclusive OR W and f	1	00 0100 dfff ffff	Z	1, 2
MOVF f, d	Move f	1	00 1000 dfff ffff	Z	1, 2
MOVWF f	Move W to f	1	00 0000 1fff ffff		
NOP -	No Operation	1	00 0000 0xx0 0000		
RLF f, d	Rotate left f through carry	1	00 1101 dfff ffff	C	1, 2
RRF f, d	Rotate right f through carry	1	00 1100 dfff ffff	C	1, 2
SUBWF f, d	Subtract W from f	1	00 0010 dfff ffff	C, DC, Z	1, 2
SWAPF f, d	Swap halves f	1	00 1110 dfff ffff		1, 2
XORWF f, d	Exclusive OR W and f	1	00 0110 dfff ffff	Z	1, 2
BIT-ORIENTED FILE REGISTER OPERATIONS					
BCF f, b	Bit Clear f	1	01 00bb bfff ffff		1, 2
BSF f, b	Bit Set f	1	01 01bb bfff ffff		1, 2
BTFSC f, b	Bit Test f, Skip if Clear	1 (2)	01 10bb bfff ffff		3
BTFSS f, b	Bit Test f, Skip if Set	1 (2)	01 11bb bfff ffff		3
LITERAL AND CONTROL OPERATIONS					
ADDLW k	Add literal to W	1	11 111x kkkk kkkk	C, DC, Z	
ANDLW k	AND literal to W	1	11 1001 kkkk kkkk	Z	
CALL k	Call subroutine	2	10 0kkk kkkk kkkk		
CLRWDT -	Clear watchdog timer	1	00 0000 0110 0100	$\overline{TO}$, $\overline{PD}$	
GOTO k	Go to address	2	10 1kkk kkkk kkkk		
IORLW k	Inclusive OR literal to W	1	11 1000 kkkk kkkk	Z	
MOVLW k	Move literal to W	1	11 00xx kkkk kkkk		
RETFIE -	Return from interrupt	2	00 0000 0000 1001		
RETLW k	Return with literal in W	2	11 01xx kkkk kkkk		
RETURN -	Return from subroutine	2	00 0000 0000 1000		
SLEEP -	Go into standby mode	1	00 0000 0110 0011	$\overline{TO}$, $\overline{PD}$	
SUBLW k	Subtract W from literal	1	11 110x kkkk kkkk	C, DC, Z	
XORLW k	Excl. OR literal to W	1	11 1010 kkkk kkkk	Z	

Notes: 1. When an I/O register is modified as a function of itself (e.g. MOVF PORTB, 1) the value used will be that value present on the pins themselves. For example, if the data latch is "1" for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2. If this instruction is executed on the TMR0 register (and, where applicable, d=1), the prescaler will be cleared if assigned to the TMR0.

3. If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.

13.1 Instruction Descriptions

ADDLW Add Literal to W

Syntax:	[*label*] ADDLW k
Operands:	$0 \le k \le 255$
Operation:	$(W) + k \rightarrow W$
Status Affected:	C, DC, Z
Encoding:	11 | 111x | kkkk | kkkk
Description:	The contents of the W register are added to the 8-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 to f

Syntax:	[*label*] ADDWF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	$(W) + (f) \rightarrow$ (dest)
Status Affected:	C, DC, Z
Encoding:	00 | 0111 | dfff | ffff
Description:	Add the contents of the W register to 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 and W

Syntax:	[*label*] ANDLW k
Operands:	$0 \le k \le 255$
Operation:	(W) .AND. (k) $\rightarrow$ W
Status Affected:	Z
Encoding:	11 | 1001 | kkkk | kkkk
Description:	The contents of W register are AND'ed with the 8-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:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(W) .AND. (f) $\rightarrow$ 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

BCF — Bit Clear f

Syntax:	[*label*] BCF f,b
Operands:	$0 \le f \le 127$ $0 \le b \le 7$
Operation:	$0 \rightarrow$ f<b>
Status Affected:	None
Encoding:	`01` `00bb` `bfff` `ffff`
Description:	Bit "b" in register "f" is reset to 0.
Words:	1
Cycles:	1

Example:	`BCF FLAG_REG, 7`
	Before Instruction
	FLAG_REG = 0xC7
	After Instruction
	FLAG_REG = 0x47

BSF — Bit Set f

Syntax:	[*label*] BSF f,b
Operands:	$0 \le f \le 127$ $0 \le b \le 7$
Operation:	$1 \rightarrow$ f<b>
Status Affected:	None
Encoding:	`01` `01bb` `bfff` `ffff`
Description:	Bit "b" in register "f" is set to 1.
Words:	1
Cycles:	1

Example:	`BSF FLAG_REG, 7`
	Before Instruction
	FLAG_REG = 0x0A
	After Instruction
	FLAG_REG = 0x8A

BTFSC — Bit Test, skip if Clear

Syntax:	[*label*] BTFSC f,b
Operands:	$0 \le f \le 127$ $0 \le b \le 7$
Operation:	skip if (f<b>) = 0
Status Affected:	None
Encoding:	`01` `10bb` `bfff` `ffff`
Description:	If bit 'b' in register 'f' is '0' then the next instruction is skipped. If bit 'b' is '0', the next instruction, fetched during the current instruction execution, is discarded and a NOP is executed instead making this a two-cycle instruction.
Words:	1
Cycles:	1(2)
Example:	see below

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

Before Instruction
PC = address `HERE`

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

BTFSS — Bit Test, skip if Set

Syntax:	[*label*] BTFSS f,b
Operands:	$0 \le f \le 127$ $0 \le b \le 7$
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 "0", the next instruction, fetched during the current instruction execution, is discarded and a NOP is executed instead making this a two-cycle instruction.
Words:	1
Cycles:	1 (2)
Example:	see below

```
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`

CALL Subroutine Call

Syntax:	[*label*] CALL k
Operands:	$0 \le k \le 2048$
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:	Subroutine call. 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 (f03). 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`

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

CLRW Clear W Register

Syntax:	[*label*] CLRW
Operands:	None
Operation:	00h $\rightarrow$ (W), 1 $\rightarrow$ Z
Status Affected:	Z
Encoding:	00 | 0001 | 0xxx | xxxx
Description:	W registered is cleared. Zero bit (Z) is set.
Words:	1
Cycles:	1
Example:	`CLRW`

Before Instruction
W = 0x5A

After Instruction
W = 0x00
Z = 1

CLRWDT Clear Watchdog Timer

Syntax:	[*label*] CLRWDT
Operands:	None
Operation:	00h $\rightarrow$ WDT, 0 $\rightarrow$ WDT prescaler, 1 $\rightarrow$ $\overline{TO}$, 1 $\rightarrow$ $\overline{PD}$
Status Affected:	$\overline{TO}$, $\overline{PD}$
Encoding:	00 | 0000 | 0110 | 0100
Description:	CLRWDT instruction resets the watchdog timer. It also resets the prescaler of the WDT. Status bits $\overline{TO}$ and $\overline{PD}$ are set.
Words:	1
Cycles:	1
Example:	`CLRWDT`

Before Instruction
WDT counter = ?

After Instruction
WDT counter = 0x00
WDT prescale = 0
$\overline{TO}$ = 0
$\overline{PD}$ = 0

COMF Complement f

Syntax:	[*label*] COMF f,d
Operands:	$0 \leq f \leq 127$ $d \in [0,1]$
Operation:	$\overline{(f)} \rightarrow$ (dest)
Status Affected:	Z
Encoding:	`00` | `1001` | `dfff` | `ffff`
Description:	The contents of register "f" are complemented. If "d" is 0 the result is stored in W. If "d" is 1 the result is stored back in register "f".
Words:	1
Cycles:	1
Example:	`COMF REG1, 0`

Before Instruction
REG1 = 0x13

After Instruction
REG1 = 0x13
W = 0xEC

DECF Decrement f

Syntax:	[*label*] DECF f,d
Operands:	$0 \leq f \leq 127$ $d \in [0,1]$
Operation:	(f)-1 $\rightarrow$ (dest)
Status Affected:	Z
Encoding:	`00` | `0011` | `dfff` | `ffff`
Description:	Decrement 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:	`DECF CNT, 1`

Before Instruction
CNT = 0x01
Z = 0

After Instruction
CNT = 0x00
Z = 1

DECFSZ Decrement f, skip if 0

Syntax:	[*label*] DECFSZ f,d
Operands:	$0 \leq f \leq 127$ $d \in [0,1]$
Operation:	(f) - 1 $\rightarrow$ d; skip if result = 0
Status Affected:	None
Encoding:	`00` | `1011` | `dfff` | `ffff`
Description:	The contents of register "f" are decremented. If "d" is 0 the result is placed in the W register. If "d" is 1 the result is placed back in register "f". If the result is 0, the next instruction, which is already fetched, is discarded. A NOP is executed instead making it a two -ycle instruction.
Words:	1
Cycles:	1 (2)

Example:
```
HERE      DECFSZ    CNT, 1
            GOTO       LOOP
CONTINUE •
         •
         •
```

Before Instruction
PC = address `HERE`

After Instruction
CNT = CNT - 1
if CNT = 0, PC = address `CONTINUE`
if CNT ≠ 0, PC = address `HERE + 1`

GOTO Unconditional Branch

Syntax:	[*label*] GOTO k
Operands:	$0 \leq k \leq 2048$
Operation:	k $\rightarrow$ PC<10:0>, (PCLATH<4:3>) $\rightarrow$ PC<12:11>
Status Affected:	None
Encoding:	`10` | `1kkk` | `kkkk` | `kkkk`
Description:	GOTO is an unconditional branch. The eleven bit immediate value is loaded into PC bits <10:0>. The upper bits of PC are loaded from PCLATH <4:3>. GOTO is a two-cycle instruction.
Words:	1
Cycles:	2
Example:	`GOTO THERE`

After Instruction
PC = Address of `THERE`

INCF — Increment f

Syntax:	[*label*] INCF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) + 1 → (dest)
Status Affected:	Z
Encoding:	00 | 1010 | dfff | ffff
Description:	The contents of register "f" are incremented. If "d" is 0 the result is placed in the W register. If "d" is 1 the result is placed back in register "f".
Words:	1
Cycles:	1
Example:	`INCF CNT, 1`

Before Instruction
CNT = 0xFF
Z = 0

After Instruction
CNT = 0x00
Z = 1

INCFSZ — Increment f, skip if 0

Syntax:	[*label*] INCFSZ f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) + 1 → (dest), skip if result = 0
Status Affected:	None
Encoding:	00 | 1111 | dfff | ffff
Description:	The contents of register "f" are incremented. If "d" is 0 the result is placed in the W register. If "d" is 1 the result is placed back in register "f". If the result is 0, the next instruction, which is already fetched, is discarded. A NOP is executed instead making it a two cycle instruction.
Words:	1
Cycles:	1 (2)
Example:	

```
HERE        INCFSZ    CNT, 1
            GOTO      LOOP
CONTINUE    •
            •
            •
```

Before Instruction
PC = address `HERE`

After Instruction
CNT = CNT + 1
if CNT = 0, PC = address `CONTINUE`
if CNT ≠ 0, PC = address `HERE + 1`

IORLW — Inclusive OR Literal with W

Syntax:	[*label*] IORLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) .OR. (k) → (W)
Status Affected:	Z
Encoding:	11 | 1000 | kkkk | kkkk
Description:	The contents of the W register are OR'ed with the eight bit literal "k". The result is placed in the W register.
Words:	1
Cycles:	1
Example:	`IORLW 0x35`

Before Instruction
W = 0x9A

After Instruction
W = 0xBF

IORWF — Inclusive OR W with f

Syntax:	[*label*] IORWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) .OR. (f) → (dest)
Status Affected:	Z
Encoding:	00 | 0100 | dfff | ffff
Description:	Inclusive OR 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:	`IORWF RESULT, 0`

Before Instruction
RESULT = 0x13
W = 0x91

After Instruction
RESULT = 0x13
W = 0x93

MOVLW Move Literal to W

Syntax:	[*label*] MOVLW k
Operands:	0 ≤ k ≤ 255
Operation:	k → (W)
Status Affected:	None
Encoding:	`11` | `00xx` | `kkkk` | `kkkk`
Description:	The 8-bit literal "k" is loaded into W register.
Words:	1
Cycles:	1
Example:	`MOVLW 0x5A`

After Instruction
W = 0x5A

MOVWF Move W to f

Syntax:	[*label*] MOVWF f
Operands:	0 ≤ f ≤ 127
Operation:	(W) → (f)
Status Affected:	None
Encoding:	`00` | `0000` | `1fff` | `ffff`
Description:	Move data from W register to register "f".
Words:	1
Cycles:	1
Example:	`MOVWF OPTION`

Before Instruction
OPTION = 0xFF
W = 0x4F

After Instruction
OPTION = 0x4F
W = 0x4F

MOVF Move f

Syntax:	[*label*] MOVF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) → (dest)
Status Affected:	Z
Encoding:	`00` | `1000` | `dfff` | `ffff`
Description:	The contents of register f is moved to destination d. If d=0, destination is W register. If d = 1, the destination is file register f itself. d = 1 is useful to test a file register since status flag Z is affected.
Words:	1
Cycles:	1
Example:	`MOVF FSR, 0`

After Instruction
W = value in FSR register

NOP No Operation

Syntax:	[*label*] NOP
Operands:	None
Operation:	No operation
Status Affected:	None
Encoding:	`00` | `0000` | `0xx0` | `0000`
Description:	No operation
Words:	1
Cycles:	1
Example:	`NOP`

OPTION **Load Option Register**

Syntax:	[*label*] OPTION
Operands:	None
Operation:	W → OPTION;
Status Affected:	None

Encoding:

00	0000	0110	0010

Description: The contents of the W register is loaded in the OPTION register. This instruction is supported for code compatibility with PIC16C5X products. Since OPTION is a readable/writable register, the user can directly address it.

Words: 1

Cycles: 1

Example:

To maintain upward compatibility with future PIC16CXX products, do not use this instruction.

RETFIE **Return from Interrupt**

Syntax:	[*label*] RETFIE
Operands:	None
Operation:	TOS → PC, 1 → GIE;
Status Affected:	None

Encoding:

00	0000	0000	1001

Description: Return from Interrupt. Stack is popped and Top of the Stack (TOS) is loaded in PC. Interrupts are enabled by setting the GIE bit. GIE is the global interrupt enable bit (INTCON<7>). This is a two-cycle instruction.

Words: 1

Cycles: 2

Example:

```
RETFIE
```

After Interrupt
PC = TOS
GIE = 1

RETLW **Return Literal to W**

Syntax:	[*label*] RETLW k
Operands:	$0 \le k \le 255$
Operation:	k → W; TOS → PC;
Status Affected:	None

Encoding:

11	01xx	kkkk	kkkk

Description: The W register is loaded with the eight bit literal "k". The program counter is loaded from the top of the stack (the return address). This is a two-cycle instruction.

Words: 1

Cycles: 2

Example:

```
         CALL    TABLE ; W contains table offset
                       ; value
         •             ; W now has table value
         •
         •
TABLE    ADDWF PC      ; W = offset
         RETLW k1      ; Begin table
         RETLW k2      ;
         •
         •
         •
         RETLW kn      ; End of table
```

Before Instruction
W = 0x07

After Instruction
W = value of k7

RETURN **Return from Subroutine**

Syntax:	[*label*] RETURN
Operands:	None
Operation:	TOS → PC;
Status Affected:	None

Encoding:

00	0000	0000	1000

Description: Return from subroutine. The stack is popped and the top of the stack (TOS) is loaded into the program counter. This is a two-cycle instruction.

Words: 1

Cycles: 2

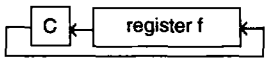
Example:

```
RETURN
```

After Interrupt
PC = TOS

RLF Rotate Left f through Carry

Syntax:	[*label*] RLF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	f<n> → d<n+1>, f<7> → C, C → d<0>;
Status Affected:	C
Encoding:	`00` `1101` `dfff` `ffff`
Description:	The contents of register "f" are rotated 1-bit to the left through the Carry Flag. If "d" is 0 the result is placed in the W register. If "d" is 1 the result is stored back in register "f".

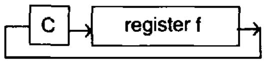


Words:	1
Cycles:	1
Example:	`RLF REG1,0`

Before Instruction
```
REG1 = 11100110
C    = 0
```
After Instruction
```
REG1 = 11100110
W    = 11001100
C    = 1
```

RRF Rotate Right f through Carry

Syntax:	[*label*] RRF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	f<n> → d<n-1>, f<0> → C, C → d<7>;
Status Affected:	C
Encoding:	`00` `1100` `dfff` `ffff`
Description:	The contents of register "f" are rotated one bit to the right through the Carry Flag. If "d" is 0 the result is placed in the W register. If "d" is 1 the result is placed back in register "f".



Words:	1
Cycles:	1
Example:	`RRF REG1,0`

Before Instruction
```
REG1 = 11100110
C    = 0
```
After Instruction
```
REG1 = 11100110
W    = 01110011
C    = 1
```

SLEEP

Syntax:	[*label*] SLEEP
Operands:	None
Operation:	00h → WDT, 0 → WDT prescaler 1 → $\overline{TO}$, 0 → $\overline{PD}$
Status Affected:	$\overline{TO}$, $\overline{PD}$
Encoding:	`00` `0000` `0110` `0011`
Description:	The Power Down status bit (PD) is cleared. Time-out status bit (TO) is set. Watchdog Timer and its prescaler are cleared. The processor is put into SLEEP mode with the oscillator stopped. See section on SLEEP mode for more details.
Words:	1
Cycles:	1
Example:	`SLEEP`

SUBLW Subtract W from Literal

Syntax:	[*label*] SUBLW k
Operands:	0 ≤ k ≤ 255
Operation:	k - (W) → (W)
Status Affected:	C, DC, Z
Encoding:	`11` `110X` `kkkk` `kkkk`
Description:	The W register is subtracted (2's complement method) from the 8-bit literal "k". The result is placed in the W register.
Words:	1
Cycles:	1
Example 1:	`SUBLW 0X02`

Before Instruction
```
W = 1
C = ?
```
After Instruction
```
W = 1
C = 1    ; result is positive
```

Example 2:

Before Instruction
```
W = 3
C = ?
```
After Instruction
```
W = FF
C = 0    ; result is negative
```

SUBWF Subtract W from f

Syntax:	[*label*] SUBWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f)-(W) → (dest)
Status Affected:	C, DC, Z
Encoding:	`00` `0010` `dfff` `ffff`
Description:	Subtract (2's complement method) the W register from 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 1:	`SUBWF REG1,1`
	Before Instruction
	REG1 = 0
	W = 1
	C = ?
	After Instruction
	REG1 = FF
	W = 1
	C = 0 ; result is negative

Example 2:	Before Instruction
	REG1 = FF
	W = 0
	C = ?
	After Instruction
	REG1 = FF
	W = 0
	C = 1 ; result is positive

TRIS Load TRIS Register

Syntax:	[*label*] TRIS f
Operands:	5 ≤ f ≤ 7
Operation:	W → TRIS register f;
Status Affected:	None
Encoding:	`00` `0000` `0110` `0fff`
Description:	This instruction is supported for code compatibility with the PIC16C5X products. Since TRIS registers are readable and writable, the user can directly address them.
Words:	1
Cycles:	1
Example:	

To maintain upward compatibility with future PIC16CXX products, do not use this instruction.

SWAPF Swap f

Syntax:	[*label*] SWAPF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	f<0:3> → d<4:7>, f<4:7> → d<0:3>;
Status Affected:	None
Encoding:	`00` `1110` `dfff` `ffff`
Description:	The upper and lower nibbles of register 'f' are exchanged. If 'd' is 0 the result is placed in W register. If 'd' is 1 the result is placed in register 'f'.
Words:	1
Cycles:	1

Example:	`SWAPF REG, 0`
	Before Instruction
	REG = 0xA5
	After Instruction
	REG = 0xA5
	W = 0x5A

XORLW Exclusive OR literal with W

Syntax:	[*label*] XORLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) .XOR. k → (W)
Status Affected:	Z
Encoding:	`11` `1010` `kkkk` `kkkk`
Description:	The contents of the W register are XOR'ed with the 8-bit literal "k". The result is placed in the W register.
Words:	1
Cycles:	1

Example:	`XORLW 0xAF`
	Before Instruction
	W = 0xB5
	After Instruction
	W = 0x1A

XORWF Exclusive OR W with f

Syntax:	[*label*] XORWF f,d
Operands:	$0 \le f \le 128$ $d \in [0,1]$
Operation:	(W) .XOR. (f) → (dest)
Status Affected	Z
Encoding:	`00` | `0110` | `dfff` | `ffff`
Description:	Exclusive OR 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:	`XORWF REG, 1`

Before Instruction
REG = 0xAF
W = 0xB5

After Instruction
REG = 0x1A
W = 0xB5

14.0 DEVELOPMENT SUPPORT

14.1 Development Tools

The PIC16C5X and PIC16CXX microcontrollers are supported with a full range of hardware and software development tools:

- PICMASTER™ Real-Time In-Circuit Emulator
- PRO MATE™ Universal Programmer
- PICSTART™ Low-Cost Prototype Programmer
- Assembler
- Software Simulator

14.2 PICMASTER™: High Performance Universal In-Circuit Emulator

The PICMASTER Universal In-Circuit Emulator is intended to provide the product development engineer with a complete microcontroller design tool set for all microcontrollers in the PIC16C5X, PIC16CXX and PIC17CXX families.

Interchangeable target probes allow the system to be easily reconfigured for emulation of different processors. The universal architecture of the PICMASTER allows expansion to support all new PIC16C5X, PIC16CXX and PIC17CXX microcontrollers.

The Emulator System is designed to operate on PC compatible machines ranging from 80286-AT® class ISA-bus systems through the new 80486 EISA-bus machines. The development software runs in the Microsoft Windows® 3.1 environment, allowing the operator access to a wide range of supporting software and accessories.

The PICMASTER has been designed as a real-time emulation system with advanced features generally found on more expensive development tools. The AT platform and Windows 3.1 environment was chosen to best make these features available to you, the end user.

The PICMASTER Emulator Universal System consists primarily of four major components:

- Host-Interface Card
- Emulator Control Pod
- Target-Specific Emulator Probe
- PC Host Emulation Control Software

The Windows 3.1 System is a multitasking operating system which will allow the developer to take full advantage of the many powerful features and functions of the PICMASTER system.

PICMASTER emulation can operate in one window, while a text editor is running in a second window.

Dynamic Data Exchange (DDE), a feature of Windows 3.1, will be available in this and future versions of the software. DDE allows data to be dynamically transferred between two or more Windows programs. With this feature, data collected with PICMASTER can be automatically transferred to a spreadsheet or database program for further analysis.

Under Windows 3.1, two or more PICMASTER emulators can run simultaneously on the same PC making development of multi-microcontroller systems possible (e.g., a system containing a PIC16Cxx processor and a PIC17Cxx processor).

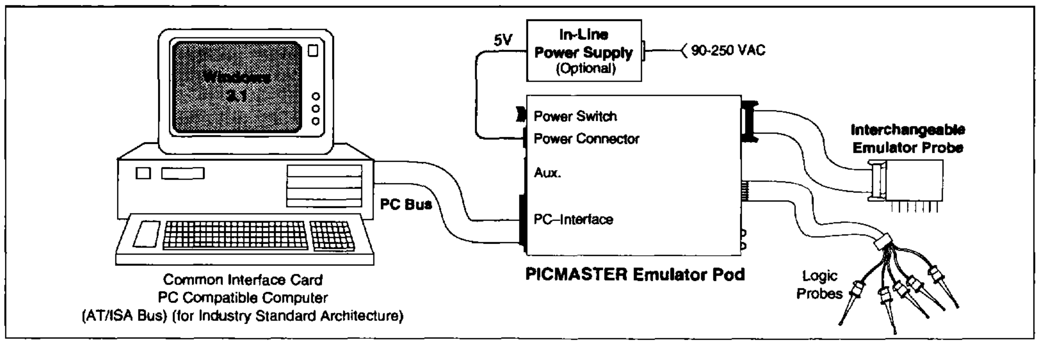
14.3 PRO MATE™: Universal Programmer

The PRO MATE Universal Programmer is a production quality programmer capable of operating in stand alone mode as well as PC-hosted mode.

The PRO MATE has programmable VDD and VPP supplies which allows it to verify programmed memory at VDD min and VDD max for maximum reliability. It has an LCD display for displaying error messages, keys to enter commands and a modular detachable socket assembly to support various package types. In stand-alone mode the PRO MATE can read, verify or program PIC16C5X, PIC16CXX and PIC17CXX devices. It can also set fuse configuration and code-protect in this mode. Its EEPROM memory holds data and parametric information even when powered down. It is ideal for low to moderate volume production.

2

FIGURE 14-1: PICMASTER SYSTEM CONFIGURATION



In PC-hosted mode, the PRO MATE connects to the PC via one of the COM (RS232) ports. A PC based user-interface software makes using the programmer simple and efficient. The user interface is full-screen and menu-based. Full screen display and editing of data, easy selection of fuse configuration and part type, easy selection of VDD min, VDD max and VPP levels, load and store to and from disk files (intel hex format) are some of the features of the software. Essential commands such as read, verify, program, blank check can be issued from the screen. Additionally, serial programming support is possible where each part is programmed with a different serial number, sequential or random.

The PRO MATE has a modular "programming socket module". Different socket modules are required for different processor types and/or package types. PRO MATE supports all PIC16C5X, PIC16CXX and PIC17CXX processors.

14.4 PICSTART™ Programmer

The PICSTART™ programmer is an easy to use, very low-cost prototype programmer. It connects to the PC via one of the COM (RS232) ports. A PC based user interface software makes using the programmer simple and efficient. The user interface is full-screen and menu-based.

14.5 Assembler (MPASM)

Cross Assembler is a PC hosted symbolic assembler. It supports all microcontroller series including the PIC16C5X CMOS, PIC16CXX, and PIC17CXX families.

MPASM offers fully featured Macro capabilities, conditional assembly, and several source and listing formats. It generates various object code formats to support Microchip's development tools as well as third party programmers.

MPASM allows full symbolic debugging from the Microchip Universal Emulator System (PICMASTER).

MPASM has the following features to assist in developing software for specific use applications.

- Provides translation of Assembler source code to object code for all Microchip microcontrollers.
- Macro Assembly Capability
- Provides Object, Listing, Symbol and special files required for debugging with one of the Microchip Emulator systems.
- Supports Hex (default), Decimal and Octal source and listing formats.

MPASM provides a full feature directive language represented by four basic classes of directives:

- **Data Directives** are those that control the allocation of memory and provide a way to refer to data items symbolically, by meaningful names.
- **Listing Directives** control the MPASM listing dis play. They allow the specification of titles and subtitles, page ejects and other listing control.
- **Control Directives** permit sections of conditionally assembled code.
- **Macro Directives** control the execution and data allocation within macro body definitions.

14.6 Software Simulator (MPSIM)

The Software Simulator allows code development in a PC host environment. It allows the user to simulate the PIC16C5X and PIC16CXX series microcontrollers on an instruction level. On any given instruction, the user may examine or modify any of the data areas or provide external stimulus to any of the pins. The input/output radix can be set by the user and the execution can be performed in single step, execute until break or in a trace mode. Two forms of symbolic debugging are available: an internal symbol table for disassembling opcodes and the displaying of source code from a listing file. The Software Simulator offers the low cost flexibility to develop and debug code outside of the laboratory environment making it an excellent multi-project software development tool.

14.7 Development Systems

For convenience, the development tools are packaged into comprehensive systems as listed in Table 14-1:

TABLE 14-1: DEVELOPMENT SYSTEM PACKAGES

Item	Name	System Description
1.	PICMASTER™ System	PICMASTER In-Circuit Emulator with your choice of Target Probe, PRO MATE Programmer, Assembler, Software Simulator and Samples.
2.	PICSTART™ System	PICSTART™ Low-Cost Prototype Programmer, Assembler, Software Simulator and Samples

14.8 Probe Specifications

The PICMASTER probes currently meet the following specifications:

PICMASTER PROBE	Devices Supported	PROBE Maximum Frequency	PROBE Operating Voltage
PROBE - 16E	PIC16C64	10 MHZ	4.5V - 5.5V

16.0 ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings †

Ambient temperature under bias	.-55 to+ 125°C
Storage Temperature	- 65°C to +150°C
Voltage on any pin with respect to Vss (except V_{DD} and $\overline{MCLR}$)	-0.6V to V_{DD} +0.6V
Voltage on V_{DD} with respect toVss	0 to +7.5 V
Voltage on $\overline{MCLR}$ with respect to Vss (Note 2)	0 to +14 V
Total power Dissipation (Note 1)	1.0 W
Maximum Current out of Vss pin	300mA
Maximum Current into V_{DD} pin	250mA
Input clamp current, I_{IK} (V_I<0 or V_I> V_{DD})	±20mA
Output clamp current, I_{OK} (V0 <0 or V0>V_{DD})	±20mA
Maximum Output Current sunk by any I/O pin	25mA
Maximum Output Current sourced by any I/O pin	25mA
Maximum Current sunk by PORTA, PORTB, and PORTE (combined)	200mA
Maximum Current sourced by PORTA, PORTB, and PORTE (combined)	200mA
Maximum Current sunk by PORTC and PORTD (combined)	200mA
Maximum Current sourced by PORTC and PORTD (combined)	200mA

Notes: 1. Power dissipation is calculated as follows: Pdis = V_{DD} x {I_{DD} - Σ Ioh} + Σ {(V_{DD}-Voh) x Ioh} + Σ(Vol x Iol)

2. Voltage spikes below Vss at the $\overline{MCLR}$ pin, inducing currents greater than 80mA, may cause latch-up. Thus, a series resistor of 50-100Ω should be used when applying a "low' level to the $\overline{MCLR}$ pin rather than pulling this pin directly to Vss.

† NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

16.1 DC CHARACTERISTICS: PIC16C64-04 (COMMERCIAL, INDUSTRIAL, AUTOMOTIVE) PIC16C64-20 (COMMERCIAL, INDUSTRIAL, AUTOMOTIVE)

DC CHARACTERISTICS	Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ + 125°C for automotive, -40°C ≤ TA ≤ + 85°C for industrial and 0°C ≤ TA ≤ +70°C for commercial Operating voltage VDD = 4.0V to 6.0V

Characteristic	Sym	Min	Typ †	Max	Units	Conditions
Supply Voltage	VDD	4.0 4.5		6.0 5.5	V V	XT, RC and LP osc configuration HS osc configuration
RAM Data Retention Voltage (Note 1)	VDR		1.5		V	Device in SLEEP mode
VDD start voltage to guarantee power on reset	VPOR		VSS		V	See section on power on reset for details
VDD rise rate to guarantee power on reset	SVDD	0.05*			V/ms	See section on power on reset for details
Supply Current (Note 2, 5)	IDD		2.7 52.5 13.5	5 105 30	mA µA mA	FOSC ≤ 4 MHz, VDD = 5.5V (Note 4) LP osc configuration FOSC = 32 KHz, VDD = 4.0V, WDT disabled HS osc configuration FOSC = 20 MHz, VDD = 5.5V (PIC16C64-20)
Power Down Current (Note 3, 5)	IPD		10.5 1.5 1.5 1.5	42 21 24 TBD	µA µA µA µA	VDD = 4.0V, WDT enabled, -40°C to +85°C VDD = 4.0V, WDT disabled, 0°C to +70°C VDD = 4.0V, WDT disabled, -40°C to +85°C VDD = 4.0V, WDT disabled, -40°C to +125°C

* These parameters are characterized but not tested.

† : Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Notes: 1. This is the limit to which VDD can be lowered in SLEEP mode without losing RAM data.

2. The supply current is mainly a function of the operating voltage and frequency. Other factors such as I/O pin loading and switching rate, oscillator type, internal code execution pattern, and temperature also have an impact on the current consumption.
The test conditions for all IDD measurements in active operation mode are:
OSC1=external square wave, from rail to rail; all I/O pins tristated, pulled to VDD, RT = VDD, $\overline{\text{MCLR}}$ = VDD; WDT enabled/disabled as specified.

3. The power down current in SLEEP mode does not depend on the oscillator type. Power down current is measured with the part in SLEEP mode, with all I/O pins in hi-impedence state and tied to VDD and VSS.

4. For RC osc configuration, current through Rext is not included. The current through the resistor can be estimated by the formula Ir = VDD/2Rext (mA) with Rext in kOhm.

5. Timer1 oscillator (when enabled) adds approximately xmA to the specification. This value is from characterization and is for design guidance only. This is not tested.

16.2 DC CHARACTERISTICS: PIC16LC64-04 (COMMERCIAL, INDUSTRIAL, AUTOMOTIVE)

DC CHARACTERISTICS

Standard Operating Conditions (unless otherwise stated)
Operating temperature -40°C ≤ TA ≤ + 125°C for automotive,
-40°C ≤ TA ≤ + 85°C for industrial and
0°C ≤ TA ≤ +70°C for commercial
Operating voltage VDD = 2.5V to 6.0V

Characteristic	Sym	Min	Typ †	Max	Units	Conditions
Supply Voltage	VDD	2.5		6.0	V	XT, RC and LP osc configuration
		4.5		5.5	V	HS osc configuration
RAM Data Retention Voltage (Note 1)	VDR		1.5		V	Device in SLEEP mode
VDD start voltage to guarantee power on reset	VPOR		Vss		V	See section on power on reset for details
VDD rise rate to guarantee power on reset	SVDD	0.05*			V/ms	See section on power on reset for details
Supply Current (Note 2, 5)	IDD		2.7	5	mA	FOSC = 4 MHz, VDD = 5.5V (Note 4)
			22.5	48	µA	FOSC = 32 KHz, VDD = 3.0V, WDT disabled
Power Down Current (Note 3, 5)	IPD		7.5	30	µA	VDD = 3.0V, WDT enabled, -40°C to +85°C
			0.9	13.5	µA	VDD = 3.0V, WDT disabled, 0°C to +70°C
			0.9	18	µA	VDD = 3.0V, WDT disabled, -40°C to +85°C
			0.9	24	µA	VDD = 3.0V, WDT disabled, -40°C to +125°C

* These parameters are characterized but not tested.
† : Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Notes:
1. This is the limit to which VDD can be lowered in SLEEP mode without losing RAM data.
2. The supply current is mainly a function of the operating voltage and frequency. Other factors such as I/O pin loading and switching rate, oscillator type, internal code execution pattern, and temperature also have an impact on the current consumption.
   The test conditions for all IDD measurements in active operation mode are:
   OSC1=external square wave, from rail to rail; all I/O pins tristated, pulled to VDD, RT = VDD, $\overline{\text{MCLR}}$ = VDD; WDT enabled/disabled as specified.
3. The power down current in SLEEP mode does not depend on the oscillator type. Power down current is measured with the part in SLEEP mode, with all I/O pins in hi-impedence state and tied to VDD and VSS.
4. For RC osc configuration, current through Rext is not included. The current through the resistor can be estimated by the formula Ir = VDD/2Rext (mA) with Rext in kOhm.
5. Timer1 oscillator (when enabled) adds approximately xmA to the specification. This value is from characterization and is for design guidance only. This is not tested.

16.3 DC CHARACTERISTICS: PIC16C64-04 (COMMERCIAL, INDUSTRIAL, AUTOMOTIVE) PIC16C64-20 (COMMERCIAL, INDUSTRIAL, AUTOMOTIVE) PIC16LC64-04 (COMMERCIAL, INDUSTRIAL, AUTOMOTIVE)

DC CHARACTERISTICS	Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +125°C for automotive, -40 ≤ TA ≤ +85°C for industrial and 0°C ≤ TA ≤ +70°C for commercial Operating voltage VDD range as described in DC spec tables 14.1/14.2					
Characteristic	**Sym**	**Min**	**Typ †**	**Max**	**Units**	**Conditions**
Input Low Voltage						
I/O ports	VIL					
- with TTL buffer		Vss		0.8 V	V	
- with Schmitt Trigger buffer		Vss		0.2 VDD	V	
MCLR, RA4/T0CKI, OSC1 (in RC mode)		Vss		0.2 VDD	V	Note 1
OSC1(in XT, HS and LP)		Vss		0.3 VDD	V	
Input High Voltage						
I/O ports	VIH					
- with TTL buffer		2.0		VDD	V	
- with Schmitt Trigger buffer		0.8 VDD		VDD		
MCLR, RA4/T0CKI		0.8 VDD		VDD	V	
OSC1 (in RC mode)		0.7 VDD		VDD	V	Note 1
PORTB weak pull-up current	IPURB	50	100	150	μA	VDD = 5V, VPIN = VSS
Input Leakage Current (Notes 2, 3)						
I/O ports	IIL			±1	μA	Vss ≤ VPIN ≤ VDD, Pin at hi-impedance
MCLR, RA4/T0CKI				±5	μA	Vss ≤ VPIN ≤ VDD
OSC1				±5	μA	Vss ≤ VPIN ≤ VDD , XT, HS and LP osc configuration
Output Low Voltage						
I/O Ports	VOL			0.6	V	IOL = 8.5 mA, VDD = 4.5V, -40°C to +85°C
				0.6	V	IOL = 7.0 mA, VDD = 4.5V, -40°C to +125°C
OSC2/CLKOUT (RC osc configuration)				0.6	V	IOL = 1.6 mA, VDD = 4.5V, -40°C to +85°C
				0.6	V	IOL = 1.2 mA, VDD = 4.5V, -40°C to +125°C
Output High Voltage						
I/O Ports (Note 3)	VOH	VDD-0.7			V	IOH = -3.0 mA, VDD = 4.5V, -40°C to +85°C
		VDD-0.7			V	IOH = -2.5 mA, VDD = 4.5V, -40°C to +125°C
OSC2/CLKOUT (RC osc configuration)		VDD-0.7			V	IOH = -1.3 mA, VDD = 4.5V, -40°C to +85°C
		VDD-0.7			V	IOH = -1.0 mA, VDD = 4.5V, -40°C to +125°C
Capacitive Loading Specs on Output Pins						
OSC2 pin	COSC2			15	pF	In XT, HS and LP modes when external clock is used to drive OSC1.
All I/O pins and OSC2 (in RC mode)	CIO			50	pF	
SCL, SDA in I²C mode	CB			400	pF	

† : Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Notes: 1. In RC oscillator configuration, the OSC1 pin is a Schmitt trigger input. It is not recommended that the PIC16C64 be driven with external clock in RC mode.

2. The leakage current on the MCLR pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.
3. Negative current is defined as coming out of the pin.
4. The user may use better of the two specs.

2

16.4 Timing Parameter Symbology

The timing parameter symbols have been created following one of the following formats:

1. TppS2ppS
2. TppS
3. Tcc:st (I²C specifications only)
4. Ts (I²C specifications only)

T			
F	Frequency	T	Time

Lowercase subscripts (pp) and their meanings:

pp			
cc	CCP1	os	OSC1
ck	CLKOUT	rd	$\overline{RD}$
cs	$\overline{CS}$	rw	$\overline{RD}$ or $\overline{WR}$
di	SDI	sc	SCK
do	SDO	ss	$\overline{SS}$
dt	Data in	t0	T0CKI
io	I/O port	t1	T1CKI
mc	$\overline{MCLR}$	wr	$\overline{WR}$

Upper case letters and their meanings:

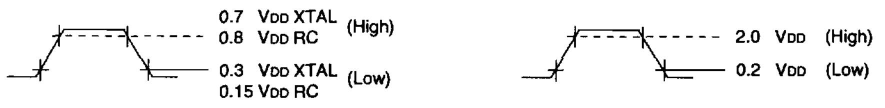
S			
F	Fall	P	Period
H	High	R	Rise
I	Invalid (Hi-impedence)	V	Valid
L	Low	Z	High Impedence
I²C only			
AA	output access	High	High
BUF	Bus free	Low	Low

Tcc:st (I²C specifications only)

CC			
HD	Hold	SU	Setup
ST			
DAT	DATA input hold	STO	STOP condition
STA	START condition		

PARAMETER MEASUREMENT INFORMATION

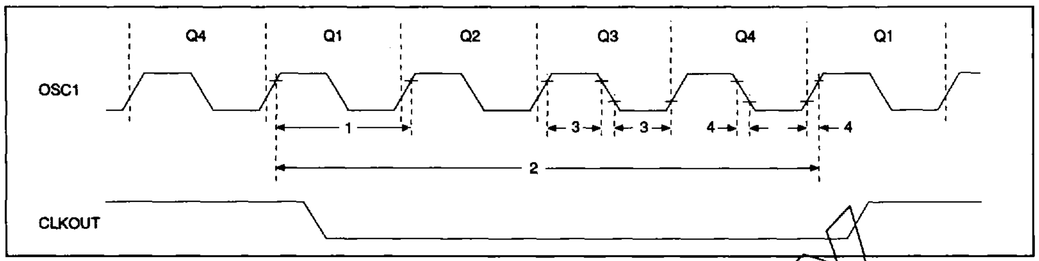
All timings are measured between high and low measurement points as indicated in the figures below.



OSC1 MEASUREMENT POINTS I/O PORT MEASUREMENT POINTS

16.5 Timing Diagrams and Specifications

External Clock Timing



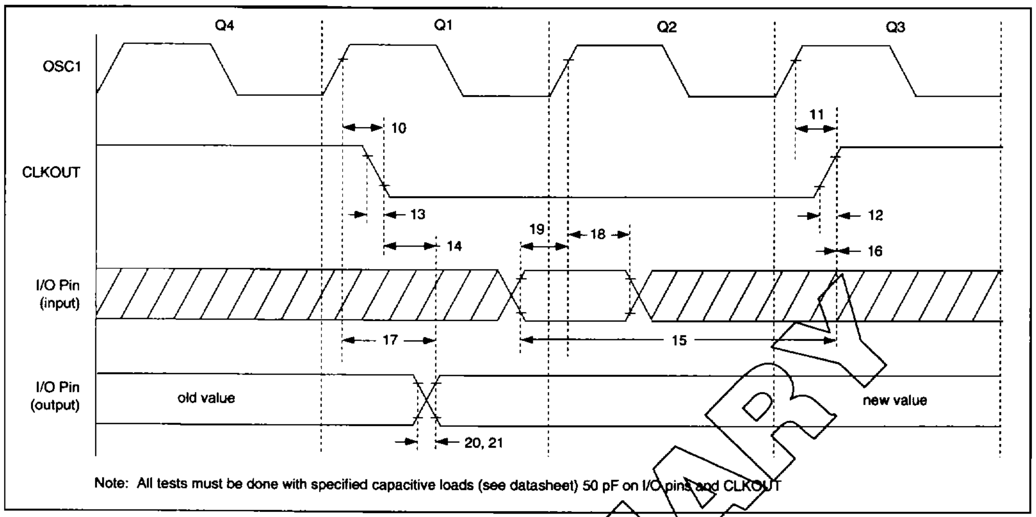
External Clock Timing Requirements

Parameter No.	Sym	Characteristic	Min	Typ †	Max	Units	Conditions
	Fos	**External CLKIN Frequency**	DC		4	MHz	XT and RC osc mode
		(Note 1)	DC		4	MHz	HS osc mode (PIC16C64-04,
				-			PIC16LC64-04)
			DC		20	MHz	HS osc mode (PIC16C64-20)
			DC		200	KHz	LP osc mode
		Oscillator Frequency	DC		4	MHz	RC osc mode
		(Note 1)	0.1		4	MHz	XT osc mode
			1		4	MHz	HS osc mode (PIC16C64-04
				-			PIC16LC64-04)
			1		20	MHz	HS osc mode (PIC16C64-20)
			DC		200	KHz	LP osc mode
1	Tos	**External CLKIN Period**	250		-	ns	XT and RC osc mode
		(Note 1)	250		-	ns	HS osc mode (PIC16C64-04,
				-			PIC16LC64-04)
			50		-	ns	HS osc mode (PIC16C64-20)
			50		-	µs	LP osc mode
		Oscillator Period	250			ns	RC osc mode
		(Note 1)	250		10,000	ns	XT osc mode
			250		1,000	ns	HS osc mode (PIC16C64-04
							PIC16LC64-04)
			50		1,000	ns	HS osc mode (PIC16C64-20)
			5			µs	LP osc mode
2	Tcy	**Instruction Cycle Time (Note 1)**	1.0	$4/F_{OS}$	DC	µs	
3	TosL,	Clock in (OSC1) High or Low Time	50	-	-	ns	XT oscillator
	TosH		2	-	-	µs	LP oscillator
			20	-	-	ns	HS oscillator
4	TosR,	Clock in (OSC1) Rise or Fall Time	25	-	-	ns	XT oscillator
	TosF		50	-	-	ns	LP oscillator
			25	-	-	ns	HS oscillator

†: Data in "Typ" column is at 5V, 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.

CLKOUT and I/O Timing



Note: All tests must be done with specified capacitive loads (see datasheet) 50 pF on I/O pins and CLKOUT

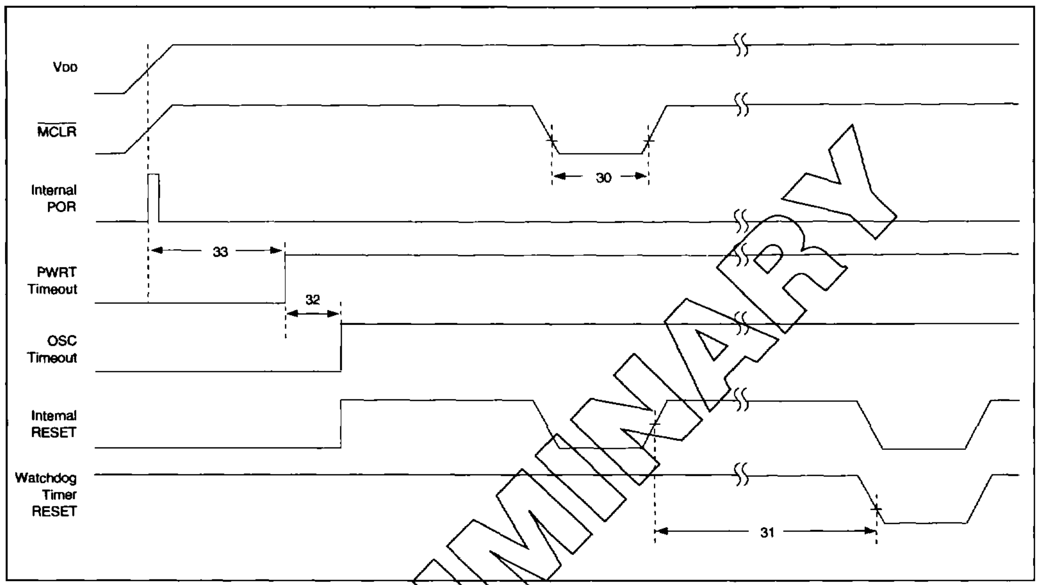
CLKOUT and I/O Timing Requirements

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
10	TosH2ckL	OSC1↑ to CLKOUT↓ (RC mode)	-	15	30	ns	
11	TosH2ckH	OSC1↑ to CLKOUT↑ (RC mode)	-	15	30	ns	
12	TckR	CLKOUT rise time (RC mode)	-	5	15	ns	
13	TckF	CLKOUT fall time (RC mode)	-	5	15	ns	
14	TckL2ioV	CLKOUT ↓ to Port out valid	-	-	0.5Tcy+20	ns	
15	TioV2ckH	Port in valid before CLKOUT ↑ (RC mode)	0.25 Tcy+25	-	-	ns	
16	TckH2ioI	Port in hold after CLKOUT ↑ (RC mode)	0	-	-	ns	
17	TosH2ioV	OSC1↑ (Q1 cycle) to Port out valid	-	-	TBD	ns	
18	TosH2ioI	OSC1↑ (Q2 cycle) to Port input invalid (I/O in hold time)	TBD	-	-	ns	
19	TioV2osH	Port input valid to OSC1↑ (I/O in setup time)	TBD	-	-	ns	
20	TioR	Port output rise time	-	10	25	ns	
21	TioF	Port output fall time	-	10	25	ns	

* These parameters are characterized but not tested.

† : Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Reset, Watchdog Timer, Oscillator Start-Up Timer and Power-Up Timer Timing



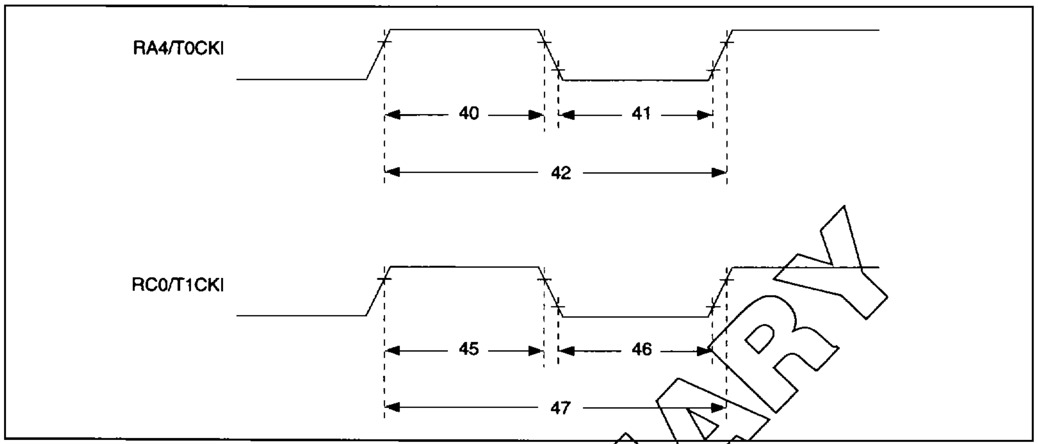
Reset, Watchdog Timer, Oscillator Start-Up Timer and Power-Up Timer Requirements

Parameter No.	Sym	Characteristic	Min	Typ †	Max	Units	Conditions
30	TmcL	MCLR Pulse Width (low)	100	-	-	ns	
31	Twdt	Watchdog Timer Timeout Period (No Prescaler)	7*	18	33*	ms	VDD = 5V, -40°C to +125°C
32	Tost	Oscillation Start-up Timer Period		1024 tosc		ms	tosc = OSC1 period
33	Tpwrt	Power-up Timer Period	28*	72	132*	ms	VDD = 5V, -40°C to +125°C

* These parameters are characterized but not tested.

† : Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Timer0 and Timer1 Clock Timings



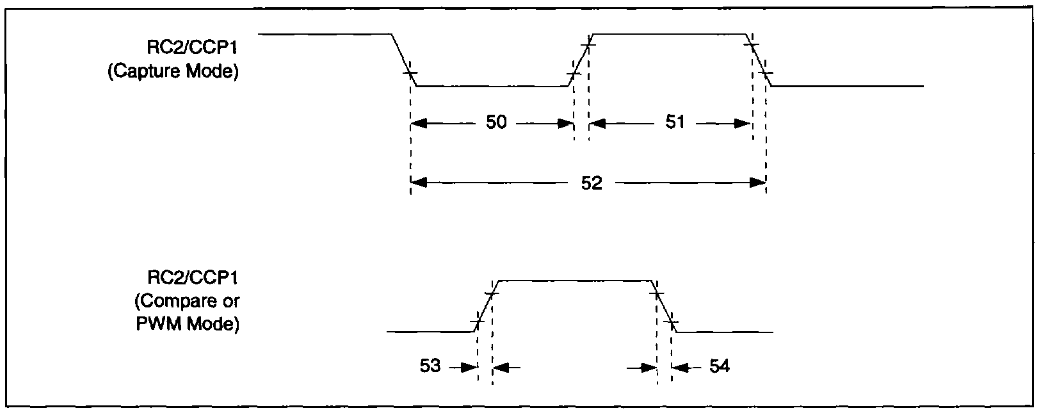
Timer0 and Timer1 Clock Requirements

Parameter No.	Sym	Characteristic		Min	Typ†	Max	Units	Conditions
40	T_{t0H}	T0CKI High Pulse Width	No Prescaler	0.5 TCY + 20*	-	-	ns	
			With Prescaler	10*	-	-	ns	
41	T_{t0L}	T0CKI Low Pulse Width	No Prescaler	0.5 TCY + 20*	-	-	ns	
			With Prescaler	10*	-	-	ns	
42	T_{t0P}	T0CKI Period		$\frac{T_{CY} + 40^*}{N}$	-	-	ns	Where N = prescale value (2, 4, ..., 256)
45	T_{t1H}	T1CKI High Time	Synchronous, No Prescaler	0.5Tcy + 20	-	-	ns	
			Synchronous, With Prescaler	10*	-	-	ns	
			Asynchronous	2 Tcy	-	-	ns	
46	T_{t1L}	T1CKI Low Time	Synchronous, No Prescaler	0.5Tcy + 20*	-	-	ns	
			Synchronous, With Prescaler	10*	-	-	ns	
			Asynchronous	2 Tcy	-	-	ns	
47	T_{t1P}	T1CKI input period	Synchronous	$\frac{T_{CY} + 40^*}{N}$	-	-	ns	N=prescale value (1, 2, 4, 8)
			Asynchronous	4 Tcy	-	-	ns	
	F_{t1}	Timer1 oscillator input frequency range (oscillator enabled by setting the T1OSCEN bit)		DC	-	200	KHz	

* These parameters are characterized but not tested.

†: Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Capture/Compare/PWM Timings



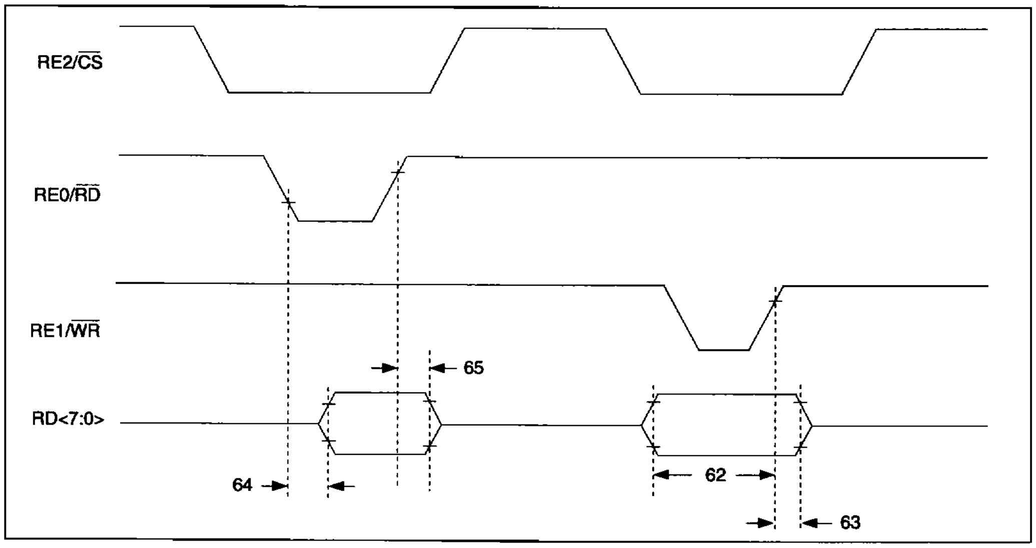
Capture/Compare/PWM Requirements

Parameter No.	Sym	Characteristic		Min	Typ†	Max	Units	Conditions
50	TccL	CCP1 input low time	No Prescaler	0.5 Tcy + 20	-	-	ns	
			With Prescaler	10	-	-	ns	
51	TccH	CCP1 input high time	No Prescaler	0.5 Tcy + 20	-	-	ns	
			With Prescaler	10	-	-	ns	
52	TccP	CCP1 input period		$\frac{Tcy + 40}{N}$	-	-	ns	N = prescale value (4 or 16)
53	TccR	CCP1 output rise time		-	10	25	ns	
54	TccF	CCP1 output fall time		-	10	25	ns	

* These parameters are characterized but not tested.

† : Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Parallel Slave Port Timing

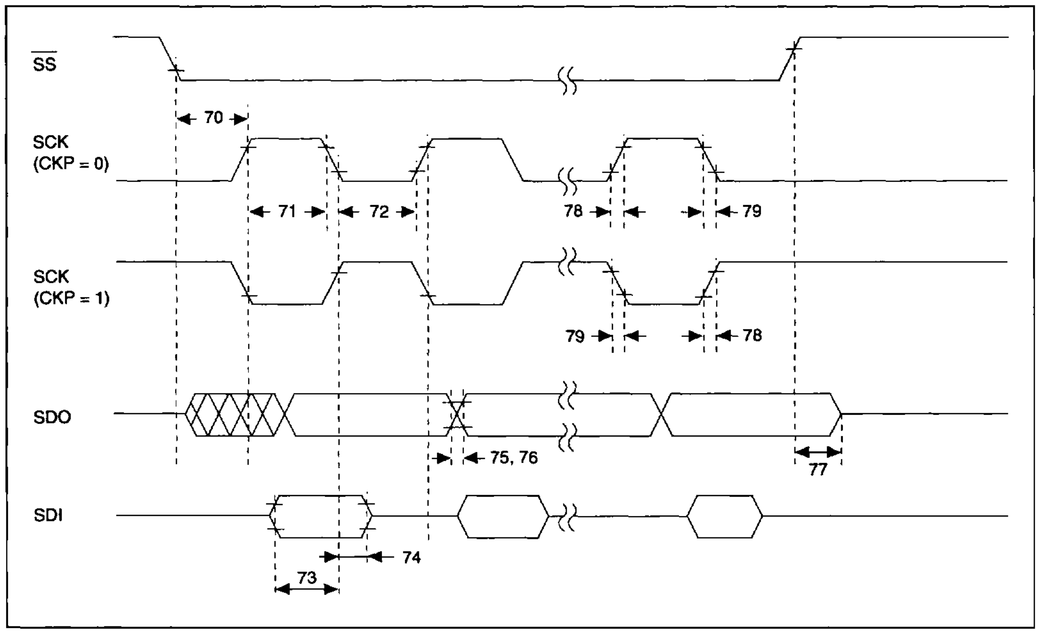

RE2/$\overline{\text{CS}}$
RE0/$\overline{\text{RD}}$
RE1/$\overline{\text{WR}}$
RD<7:0>
65
64
62
63

Parallel Slave Port Requirements

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
62	TdtV2wrH	Data in valid before $\overline{\text{WR}}$↑ or $\overline{\text{CS}}$↑ (setup time)	20	-	-	ns	
63	TwrH2dtI	$\overline{\text{WR}}$↑ or $\overline{\text{CS}}$↑ to data–in invalid (hold time)	20	-	-	ns	
64	TrdL2dtV	$\overline{\text{RD}}$↓ and $\overline{\text{CS}}$↓ to data–out valid	-	-	40	ns	
65	TrdH2dtI	$\overline{\text{RD}}$↑ or $\overline{\text{CS}}$↓ to data–out invalid	10	-	30	ns	

† : Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

SPI Mode Timing

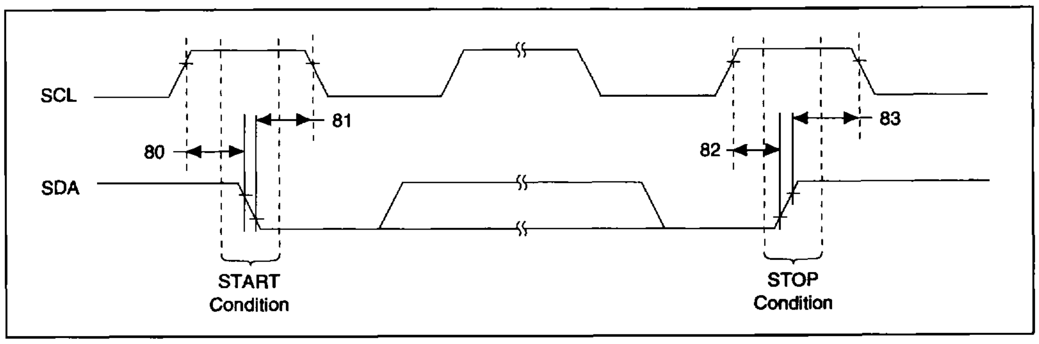


SPI Mode Requirements

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
70	TssL2scH, TssL2scL,	$\overline{SS}$↓ to SCK↓ or SCK↑ input	Tcy	-	-	ns	
71	TscH	SCK input high time (slave mode)	Tcy + 20	-	-	ns	
72	TscL	SCK input low time (slave mode)	Tcy + 20	-	-	ns	
73	TdiV2scH, TdiV2scL	SDI data input valid before SCK edge	Tcy	-	-	ns	
74	TscH2dil, TscL2dil,	SDI data input invalid after SCK edge	0.5Tcy	-	-	ns	
75	TdoR	SDO data output rise time	-	10	25	ns	
76	TdoF	SDO data output fall time	-	10	25	ns	
77	TssH2doZ	$\overline{SS}$↓ to SDO output hi-impedence	10	-	50	ns	
78	TscR	SCK output rise time (master mode)	-	10	25	ns	
79	TscF	SCK output fall time (master mode)	-	10	25	ns	

† : Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

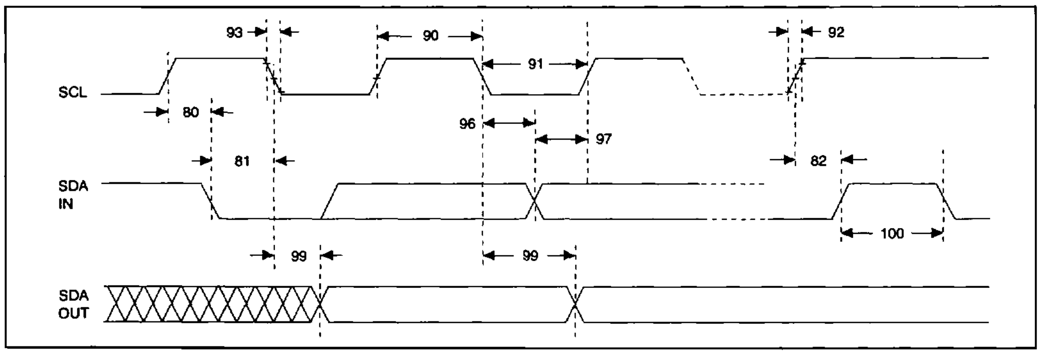
I²C Bus Start/Stop Bits Timing



I²C Bus Start/Stop Bits Requirements

Parameter No.	Sym	Characteristic		Min	Typ	Max	Units	Conditions
80	TSU:STA	START condition	100 KHZ mode	4700	-	-	ns	Only relevant for repeated
		Setup time	400 KHz mode	600	-	-		START condition
81	THD:STA	START condition	100 KHz mode	4000	-	-	ns	After this period the
		Hold time	400 KHz mode	600	-	-		first clock pulse is generated
82	TSU:STO	STOP condition	100 KHZ mode	4700	-	-	ns	
		Setup time	400 KHz mode	600	-	-		
83	THD:STO	STOP condition	100 KHz mode	4000	-	-	ns	
		Hold time	400 KHz mode	600	-	-		

I²C Bus Data Timing



I²C Bus Data Requirements

Parameter No.	Sym	Characteristic		Min	Max	Units	Conditions
90	THIGH	Clock high time	100 KHz mode	4.0	-	µs	PIC16C64 must operate at a minimum of 1.5 MHz
			400 KHz mode	0.6	-	µs	PIC16C64 must operate at a minimum of 10 MHz
			SSP Module	1.5 Tcy	-		
91	TLOW	Clock low time	100 KHz mode	4.7	-	µs	PIC16C64 must operate at a minimum of 1.5 MHz
			400 KHz mode	1.3	-	µs	PIC16C64 must operate at a minimum of 10 MHz
			SSP Module	1.5 Tcy	-		
92	TR	SDA and SCL rise time	100 KHz mode	-	1000	ns	
			400 KHz mode	20+0.1 Cb	300	ns	Cb is specified to be from 10-400 pF
93	TF	SDA and SCL fall time	100 KHz mode	-	300	ns	
			400 KHz mode	20+0.1 Cb	300	ns	Cb is specified to be from 10-400 pF
80	TSU:STA	START condition setup time	100 KHz mode	4.7	-	µs	Only relevant for repeated START condition
			400 KHz mode	0.6	-	µs	
81	THD:STA	START condition hold time	100 KHz mode	4.0	-	µs	After this period the first clock pulse is generated
			400 KHz mode	0.6	-	µs	
96	THD:DAT	Data input hold time	100 KHz mode	0	-	ns	
			400 KHz mode	0	0.9	µs	
97	TSU:DAT	Data input setup time	100 KHz mode	250	-	ns	
			400 KHz mode	100	-	ns	Note 2
82	TSU:STO	STOP condition setup time	100 KHz mode	4.7	-	µs	
			400 KHz mode	0.6		µs	
99	TAA	Output valid from clock	100 KHz mode	300	3500	ns	Note 1
			400 KHz mode	-	-	-	
100	TBUF	Bus free time	100 KHz mode	4.7	-	µs	Time the bus must be free before a new transmission can start
			400 KHz mode	1.3	-	µs	
	Cb	Bus capacitive loading		-	400	pF	

Note 1: As a transmitter, the device must provide this internal minimum delay time to bridge the undefined region (min. 300ns) of the falling edge of SCL to avoid unintended generation of START or STOP conditions.

2: A fast-mode I²C-bus device can be used in a standard-mode I²C-bus system, but the requirement tsu;DAT≥250ns must then be met. This will automatically be the case if the device does not stretch the LOW period of the SCL signal. If such a device does stretch the LOW period of the SCL signal, it must output the next data bit to the SDA line t_R max.+tsu;DAT=1000+250=1250ns (according to the standard-mode I²C bus specification) before the SCL line is released.

17.0 DC AND AC CHARACTERISTICS (GRAPHS/TABLES)

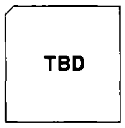
NOT AVAILABLE AT THIS TIME

20.0 PACKAGING INFORMATION

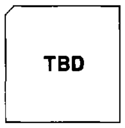
See Section 11 of the Data Book.

20.1 Package Marking Information

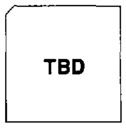
44L PLCC



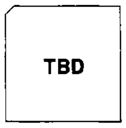
Example



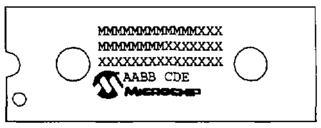
44L PQFP



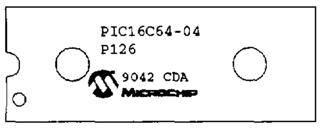
Example



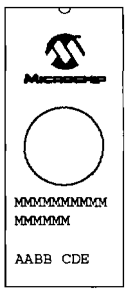
40L PDIP (.600 mil)



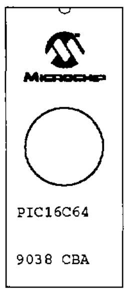
Example



40L Cerdip



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 or origin in which part was assmebled

Note: In the event the full Microchip part number can not 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.

APPENDIX A

The following are the list of modifications over the PIC16C5X microcontroller family:

1. Instruction word length is increased to 14-bit. This allows larger page sizes both in program memory (2K now as opposed to 512 before) and register file (128 bytes now versus 32-bytes before).
2. A PC high latch register (PCLATH) is added to handle program memory paging. PA2, PA1, PA0 bits are removed from status register.
3. Data memory paging is redefined slightly. Status register is modified.
4. Four new instructions have been added: RETURN, RETFIE, ADDLW, and SUBLW.

   Two instructions TRIS and OPTION are being phased out although they are kept for compatibility with PIC16C5X.
5. OPTION and TRIS registers are made addressible.
6. Interrupt capability is added. Interrupt vector is at 0004h.
7. Stack size is increased to eight deep.
8. Reset vector is changed to 0000h.
9. Reset of all registers is revisited. Five different reset (and wake-up) types are recognized. Registers are reset differently.
10. Wake up from SLEEP through interrupt is added.
11. Two separate timers oscillator start-up timer (OST) and power-up timer (PWRT) are included for more reliable power-up. These timers are invoked selectively to avoid unnecessary delays on power-up and wake-up.
12. PORTB has weak pull-ups and interrupt on change feature.
13. RTCC pin is also a port pin (RA4) now.
14. FSR is made a full eight bit register.
15. "In system programming" is made possible. The user can program PIC16CXX devices using only five pins: VDD, Vss, MCLR/VPP, RB6 (clock) and RB7 (data in/out).
16. PCON status register is added with a Power-On Reset (POR) status bit.
17. Code protection scheme is enhanced such that portions of the program memory can be protected, while the remainder is unprotected.

APPENDIX B

To convert code written for PIC16C5X to PIC16CXX, the user should take the following steps:

1. Remove any program memory page select operations (PA2, PA1, PA0 bits) for CALL, GOTO.
2. Revisit any computed jump operations (write to PC or add to PC, etc.) to make sure page bits are set properly under the new scheme.
3. Eliminate any data memory page switching. Redefine data variables to reallocate them.
4. Verify all writes to STATUS, OPTION, and FSR registers since these have changed.
5. Change reset vector to 0000h.

Index

Preliminary

Table of Examples

Table of Tables

Table of Figures

CONNECTING TO MICROCHIP BBS

Connect world wide to the Microchip BBS using the CompuServe communications network. In most cases a local call is your only expense. The Microchip BBS connection does not use CompuServe membership services, therefore **you do not need CompuServe membership to join Microchip's BBS.**

The procedure to connect will vary slightly from country to country. Please check with your local CompuServe agent for details if you have a problem. CompuServe services allows multiple users at baud rates up to 9600.

To connect:

1. Set your modem to 8-bit, No parity, and One stop (8N1). This is not the normal CompuServe setting which is 7E1.
2. Dial your local CompuServe phone number.
3. Depress **<ENTER>** and a garbage string will appear because CompuServe is expecting a 7E1 setting.
4. Type + **<ENTER>** and `Host Name:` will appear.
5. Type **MCHIPBBS<ENTER>** and you will be connected to the Microchip BBS.

In the United States, to find CompuServe's phone number closest to you, set your modem to 7E1 and dial (800) 848-4480 for 300-2400 baud or (800) 331-7166 for 9600 baud connection. After the system responds with `Host Name:`, type

NETWORK<ENTER> and follow CompuServe's directions.

For voice information (or calling from overseas), you may call (614) 457-1550 for your local CompuServe number.

PIC16C64 Product Identification System

To order or to obtain information, e.g., on pricing or delivery, please use the listed part numbers, and refer to the factory or the listed sales offices.

PART NO. -XX X /XX XXX

Pattern:	3-Digit Pattern Code for QTP/SQTP (factory specified) (blank for OTP and Windowed Parts)	
Package:	P	= PDIP (600 mil)
	L	= PLCC
	PQ	= MQFP (Metric PQFP)
Temperature Range:	-	= 0˚ C to +70˚C (T for tape/reel)
	I	= -40˚ C to +85˚C (S for tape/reel)
	E	= -40˚C to +125˚C
Frequency Range:	04	= 4 MHz
	10	= 10 MHz
	20	= 20 MHz
Device:	PIC16C64	: Standard VDD range
	PIC16LC64	: Extended VDD range

Examples:

a) PIC16C64 - 04/P = Commercial temp., DIP (600 mil) package, 4 MHz, standard VDD limits, OTP part

b) PIC16C64 - 04I/PQ462 = Industrial temp., MQFP package, 4 MHz, standard VDD limits, pattern #462 (QTP or SQTP parts)

Sales and Support

Products supported by a preliminary Data Sheet may possibly 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 (see below)
2. The Microchip Corporate Literature Center U.S. FAX: (602) 786-7277
3. The Microchip's Bulletin Board, via your local Compuserve number.

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

For latest version information and upgrade kits for Microchip Development Tools, please call 1-800-755-2345 or 1-602-786-7302.