PIC16C745/765

8-Bit CMOS Microcontrollers with USB

Devices included in this data sheet:

- PIC16C745
- PIC16C765

Microcontroller Core Features:

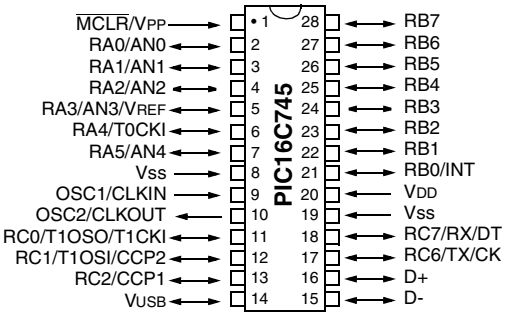
- High-performance RISC CPU
- Only 35 single word instructions

Device	Memory		Pins	A/D Resolution	A/D Channels
	Program x14	Data x8			
PIC16C745	8K	256	28	8	5
PIC16C765	8K	256	40	8	8

- All single cycle instructions except for program branches which are two cycle
- Interrupt capability (up to 12 internal/external interrupt sources)
- Eight level deep hardware stack
- Direct, indirect and relative addressing modes
- Power-on Reset (POR)
- Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Brown-out detection circuitry for Brown-out Reset (BOR)
- Programmable code-protection
- Power saving SLEEP mode
- Selectable oscillator options
  - EC - External clock (24 MHz)
  - E4 - External clock with PLL (6 MHz)
  - HS - Crystal/Resonator (24 MHz)
  - H4 - Crystal/Resonator with PLL (6 MHz)
- Processor clock of 24 MHz derived from 6 MHz crystal or resonator
- Fully static low-power, high-speed CMOS
- In-Circuit Serial Programming™ (ICSP)
- Operating voltage range
  - 4.35 to 5.25V
- High Sink/Source Current 25/25 mA
- Wide temperature range
  - Industrial (-40°C - 85°C)
- Low-power consumption:
  - ~ 16 mA @ 5V, 24 MHz
  - 100 μA typical standby current

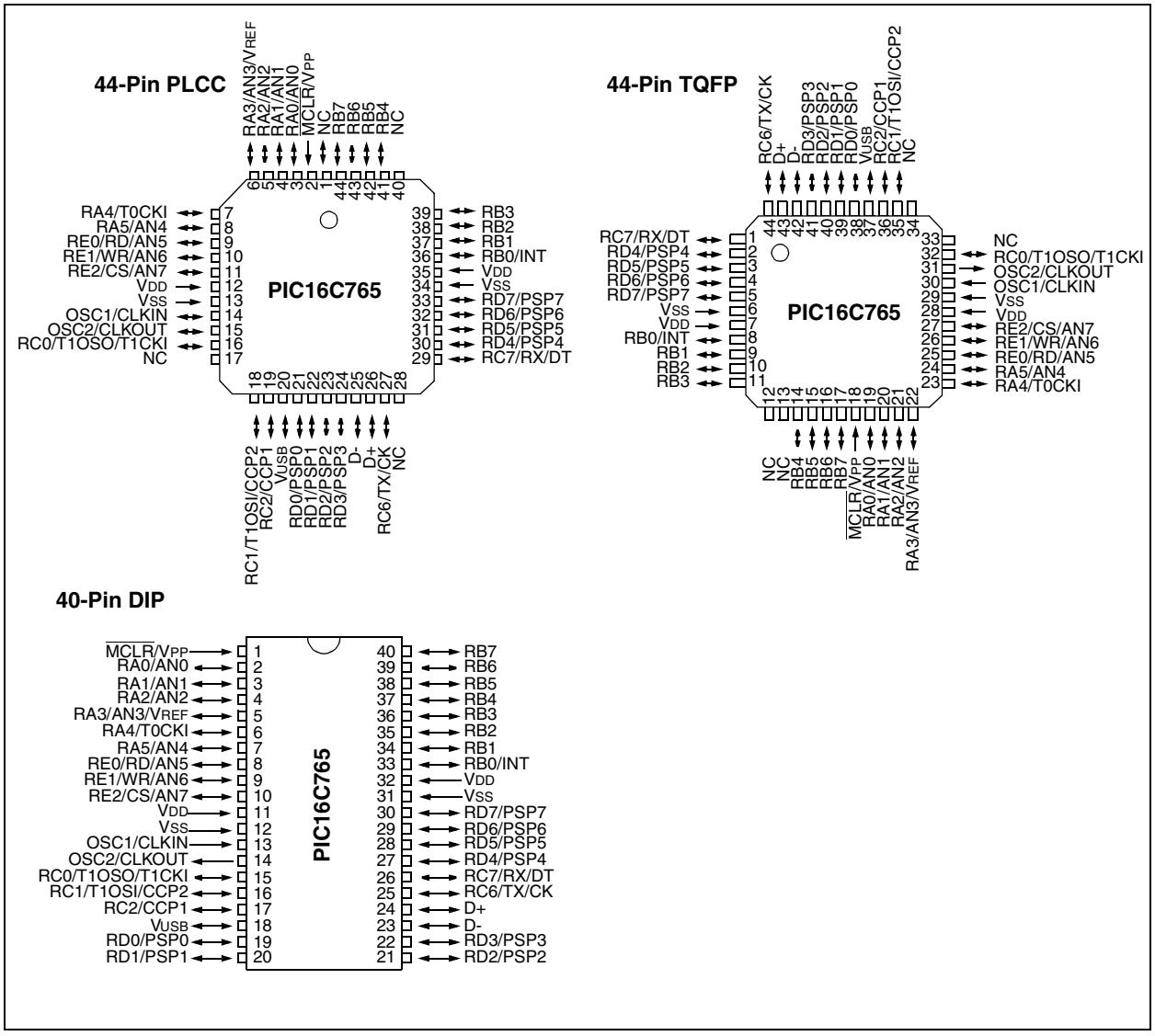
Pin Diagrams

28-Pin DIP, SOIC



Peripheral Features:

- Universal Serial Bus (USB 1.1)
  - Soft attach/detach
- 64 bytes of USB dual port RAM
- 22 (PIC16C745) or 33 (PIC16C765) I/O pins
  - Individual direction control
  - 1 high voltage open drain (RA4)
  - 8 PORTB pins with:
    - Interrupt-on-change control (RB<7:4> only)
    - Weak pull-up control
  - 3 pins dedicated to USB
- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler can be incremented during SLEEP via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- 2 Capture, Compare and PWM modules
  - Capture is 16-bit, max. resolution is 10.4 ns
  - Compare is 16-bit, max. resolution is 167 ns
  - PWM maximum resolution is 10-bit
- 8-bit multi-channel Analog-to-Digital converter
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI)
- Parallel Slave Port (PSP) 8-bits wide, with external $\overline{RD}$, $\overline{WR}$ and $\overline{CS}$ controls (PIC16C765 only)



Key Features PICmicro™ Mid-Range Reference Manual (DS33023)	PIC16C745	PIC16C765
Operating Frequency	6 MHz or 24 MHz	6 MHz or 24 MHz
Resets (and Delays)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)
Program Memory (14-bit words)	8K	8K
Data Memory (bytes)	256	256
Dual Port Ram	64	64
Interrupt Sources	11	12
I/O Ports	22 (Ports A, B, C)	33 (Ports A, B, C, D, E)
Timers	3	3
Capture/Compare/PWM modules	2	2
Analog-to-Digital Converter Module	5 channel x 8 bit	8 channel x 8 bit
Parallel Slave Port	—	Yes
Serial Communication	USB, USART/SCI	USB, USART/SCI
Brown-out Detect Reset	Yes	Yes

Table of Contents

1.0 General Description 5
2.0 PIC16C745/765 Device Varieties 7
3.0 Architectural Overview 9
4.0 Memory Organization 15
5.0 I/O Ports 31
6.0 Timer0 Module 43
7.0 Timer1 Module 45
8.0 Timer2 Module 49
9.0 Capture/Compare/PWM Modules 51
10.0 Universal Serial Bus 57
11.0 Universal Synchronous Asynchronous Receiver Transmitter (USART) 77
12.0 Analog-to-Digital Converter (A/D) Module 91
13.0 Special Features of the CPU 99
14.0 Instruction Set Summary 113
15.0 Development Support 121
16.0 Electrical Characteristics 127
17.0 DC and AC Characteristics Graphs and Tables 145
18.0 Packaging Information 147
Index 157
On-Line Support 161
Reader Response 162
Product Identification System 163

To Our Valued Customers

Most Current Data Sheet

To obtain the most up-to-date version of this data sheet, please register at our Worldwide Web site at:

http://www.microchip.com

You can determine the version of a data sheet by examining its literature number found on the bottom outside corner of any page. The last character of the literature number is the version number. e.g., DS30000A is version A of document DS30000.

New Customer Notification System

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

Errata

An errata sheet may exist for current devices, describing minor operational differences (from the data sheet) and recommended workarounds. As device/documentation issues become known to us, we will publish an errata sheet. The errata will specify the revision of silicon and revision of document to which it applies.

To determine if an errata sheet exists for a particular device, please check with one of the following:

- Microchip's Worldwide Web site; http://www.microchip.com
- Your local Microchip sales office (see last page)
- The Microchip Corporate Literature Center; U.S. FAX: (480) 786-7277

When contacting a sales office or the literature center, please specify which device, revision of silicon and data sheet (include literature number) you are using.

Corrections to this Data Sheet

We constantly strive to improve the quality of all our products and documentation. We have spent a great deal of time to ensure that this document is correct. However, we realize that we may have missed a few things. If you find any information that is missing or appears in error, please:

- Fill out and mail in the reader response form in the back of this data sheet.
- E-mail us at webmaster@microchip.com.

We appreciate your assistance in making this a better document.

NOTES:

1.0 GENERAL DESCRIPTION

The PIC16C745/765 devices are low cost, high-performance, CMOS, fully-static, 8-bit microcontrollers in the PIC16CXX mid-range family.

All PIC® microcontrollers employ an advanced RISC architecture. The PIC16C745/765 microcontroller family has enhanced core features, eight-level deep stack and multiple internal and external interrupt sources. The separate instruction and data buses of the Harvard architecture allow a 14-bit wide instruction word with the separate 8-bit wide data. The two stage instruction pipeline allows all instructions to execute in a single cycle, except for program branches, which require two cycles. A total of 35 instructions (reduced instruction set) are available. Additionally, a large register set gives some of the architectural innovations used to achieve a very high performance.

The PIC16C745 device has 22 I/O pins. The PIC16C765 device has 33 I/O pins. Each device has 256 bytes of RAM. In addition, several peripheral features are available including: three timer/counters, two Capture/Compare/PWM modules and two serial ports. The Universal Serial Bus (USB 1.1) low speed peripheral provides bus communications. The Universal Synchronous Asynchronous Receiver Transmitter (USART) is also known as the Serial Communications Interface or SCI. Also, a 5-channel high-speed 8-bit A/D is provided on the PIC16C745, while the PIC16C765 offers 8 channels. The 8-bit resolution is ideally suited for applications requiring a low cost analog interface (e.g., thermostat control, pressure sensing, etc.).

The PIC16C745/765 devices have special features to reduce external components, thus reducing cost, enhancing system reliability and reducing power consumption. There are 4 oscillator options, of which EC is for the external regulated clock source, E4 is for the external regulated clock source with the PLL enabled, HS is for the high speed crystals/resonators and H4 is for high speed crystals/resonators with the PLL enabled. The SLEEP (power-down) feature provides a power-saving mode. The user can wake-up the chip from SLEEP through several external and internal interrupts and RESETS.

A highly reliable Watchdog Timer (WDT), with a dedicated on-chip RC oscillator, provides protection against software lock-up, and also provides one way of waking the device from SLEEP.

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

The PIC16C745/765 devices fit nicely in many applications ranging from security and remote sensors to appliance controls and automotives. The EPROM technology makes customization of application programs (data loggers, industrial controls, UPS) extremely fast and convenient. The small footprint packages make this microcontroller series perfect for all applications with space limitations. Low-cost, low-power, high-performance, ease of use and I/O flexibility make the PIC16C745/765 devices very versatile, even in areas where no microcontroller use has been considered before (e.g., timer functions, serial communication, capture and compare, PWM functions and coprocessor applications).

1.1 Family and Upward Compatibility

Users familiar with the PIC16C5X microcontroller family will realize that this is an enhanced version of the PIC16C5X architecture. Code written for the PIC16C5X can be easily ported to the PIC16C745/765 family of devices.

1.2 Development Support

PIC® devices are supported by the complete line of Microchip Development tools.

Please refer to Section 15.0 for more details about Microchip's development tools.

NOTES:

2.0 PIC16C745/765 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 in the PIC16C745/765 Product Identification System section at the end of this data sheet. When placing orders, please use that page of the data sheet to specify the correct part number.

2.1 UV Erasable Devices

The UV erasable version, offered in windowed CERDIP packages, is optimal for prototype development and pilot programs. This version can be erased and reprogrammed to any of the supported oscillator modes.

Microchip's PICSTART® Plus and PRO MATE® II programmers both support programming of the PIC16C745/765.

2.2 One-Time-Programmable (OTP) Devices

The availability of OTP devices is especially useful for customers who need the flexibility for frequent code updates and small volume applications.

The OTP devices, packaged in plastic packages, permit the user to program them once. In addition to the program memory, the configuration bits must also 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 choose 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 configuration options already programmed by the factory. Certain code and prototype verification procedures apply before production shipments are available. Please contact your local Microchip Technology sales office for more details.

2.4 Serialized Quick-Turnaround Production (SQTP℠) Devices

Microchip offers a 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.

NOTES:

3.0 ARCHITECTURAL OVERVIEW

The high performance of the PIC16C745/765 family can be attributed to a number of architectural features commonly found in RISC microprocessors. To begin with, the PIC16C745/765 uses a Harvard architecture, in which program and data are accessed from separate memories using separate buses. This improves bandwidth over traditional von Neumann architecture in which program and data are fetched from the same memory using the same bus. Separating program and data buses further allows instructions to be sized differently than the 8-bit wide data word. Instruction opcodes are 14-bits 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 (Example 3-1). Consequently, most instructions execute in a single cycle (166.6667 ns @ 24 MHz) except for program branches.

Device	Memory		Pins	A/D Resolution	A/D Channels
	Program x14	Data x8			
PIC16C745	8K	256	28	8	5
PIC16C765	8K	256	40	8	8

The PIC16C745/765 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 PIC16C745/765 has an 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 PIC16C745/765 simple yet efficient. In addition, the learning curve is reduced significantly.

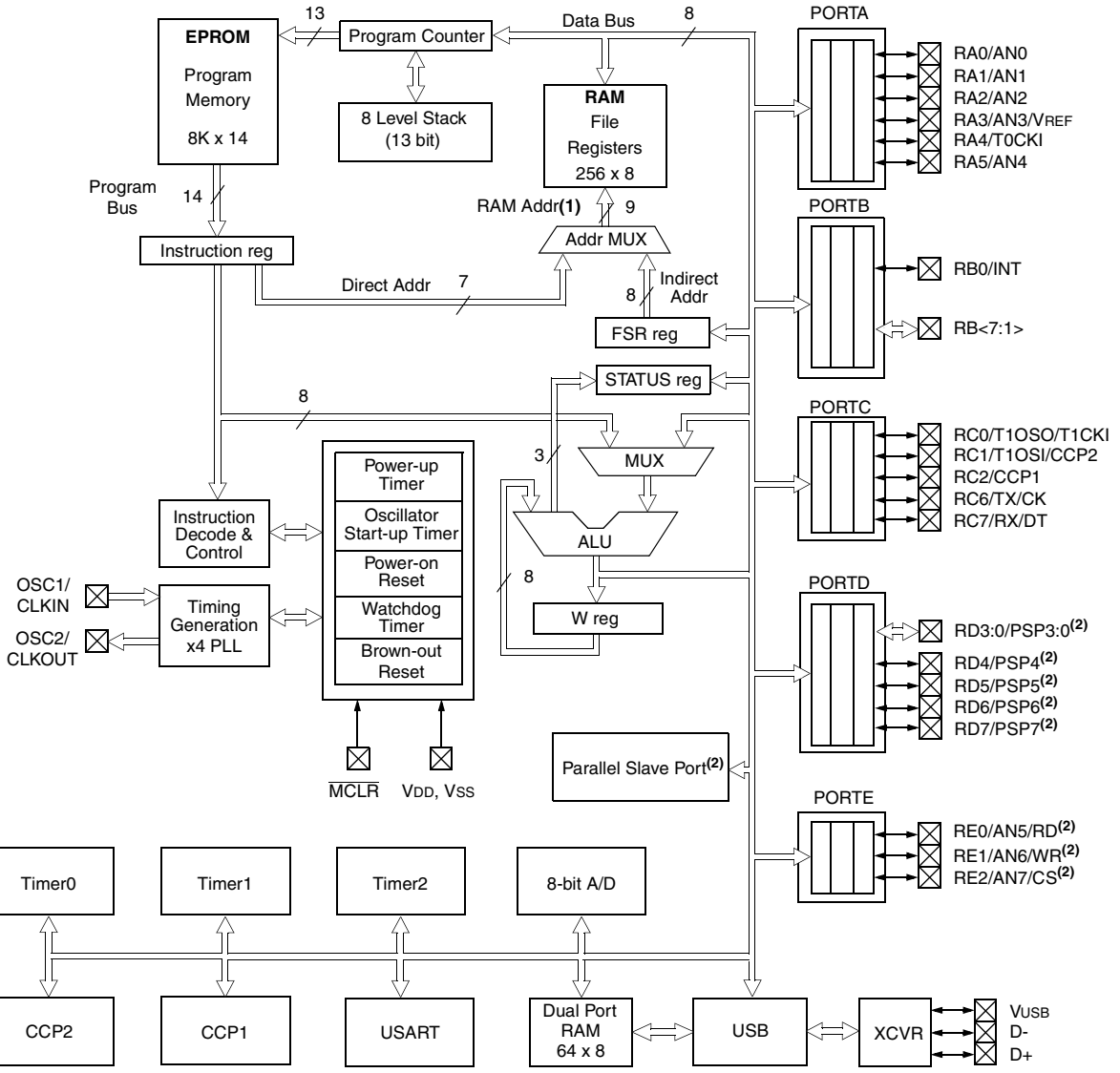
PIC16C745/765 devices contain an 8-bit ALU and working register. The ALU is a general purpose arithmetic unit. It performs arithmetic and Boolean functions between the data in the working register and any register file.

The ALU is 8-bits 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 borrow bit and a digit borrow out bit, respectively, in subtraction. See the `SUBLW` and `SUBWF` instructions for examples.

FIGURE 3-1: PIC16C745/765 BLOCK DIAGRAM



Note 1: Higher order bits are from the STATUS register.
2: Not available on PIC16C745.

TABLE 3-1: PIC16C745/765 PINOUT DESCRIPTION

Name	Function	Input Type	Output Type	Description
$\overline{MCLR}$/VPP	$\overline{MCLR}$	ST	—	Master Clear
	VPP	Power	—	Programming Voltage
OSC1/CLKIN	OSC1	Xtal	—	Crystal/Resonator
	CLKIN	ST	—	External Clock Input
OSC2/CLKOUT	OSC2	—	Xtal	Crystal/Resonator
	CLKOUT	—	CMOS	Internal Clock (FINT/4) Output
RA0/AN0	RA0	ST	CMOS	Bi-directional I/O
	AN0	AN	—	A/D Input
RA1/AN1	RA1	ST	CMOS	Bi-directional I/O
	AN1	AN	—	A/D Input
RA2/AN2	RA2	ST	CMOS	Bi-directional I/O
	AN2	AN	—	A/D Input
RA3/AN3/VREF	RA3	ST	CMOS	Bi-directional I/O
	AN3	AN	—	A/D Input
	VREF	AN	—	A/D Positive Reference
RA4/T0CKI	RA4	ST	OD	Bi-directional I/O
	T0CKI	ST	—	Timer 0 Clock Input
RA5/AN4	RA5	ST		Bi-directional I/O
	AN4	AN	—	A/D Input
RB0/INT	RB0	TTL	CMOS	Bi-directional I/O(1)
	INT	ST	—	Interrupt
RB1	RB1	TTL	CMOS	Bi-directional I/O(1)
RB2	RB2	TTL	CMOS	Bi-directional I/O(1)
RB3	RB3	TTL	CMOS	Bi-directional I/O(1)
RB4	RB4	TTL	CMOS	Bi-directional I/O with Interrupt-on-Change(1)
RB5	RB5	TTL	CMOS	Bi-directional I/O with Interrupt-on-Change(1)
RB6/ICSPC	RB6	TTL	CMOS	Bi-directional I/O with Interrupt-on-Change(1)
	ICSPC	ST		In-Circuit Serial Programming Clock Input
RB7/ICSPD	RB7	TTL	CMOS	Bi-directional I/O with Interrupt-on-Change(1)
	ICSPD	ST	CMOS	In-Circuit Serial Programming Data I/O
RC0/T1OSO/T1CKI	RC0	ST	CMOS	Bi-directional I/O
	T1OSO	—	Xtal	T1 Oscillator Output
	T1CKI	ST	—	T1 Clock Input
RC1/T1OSI/CCP2(1)	RC1	ST	CMOS	Bi-directional I/O
	T1OSI	Xtal	—	T1 Oscillator Input
	CCP2			Capture In/Compare Out/PWM Out 2
RC2/CCP1	RC2	ST	CMOS	Bi-directional I/O
	CCP1			Capture In/Compare Out/PWM Out 1
VUSB	VUSB		Power	Regulator Output Voltage
D-	D-	USB	USB	USB Differential Bus
D+	D+	USB	USB	USB Differential Bus

Legend: OD = open drain, ST = Schmitt Trigger

Note 1: Weak pull-ups. PORT B pull-ups are byte wide programmable.

2: PIC16C765 only.

TABLE 3-1: PIC16C745/765 PINOUT DESCRIPTION (CONTINUED)

Name	Function	Input Type	Output Type	Description
RC6/TX/CK	RC6	ST	CMOS	Bi-directional I/O
	TX	—	CMOS	USART Async Transmit
	CK	ST	CMOS	USART Master Out/Slave In Clock
RC7/RX/DT	RC7	ST	CMOS	Bi-directional I/O
	RX	ST	—	USART Async Receive
	DT	ST	CMOS	USART Data I/O
RD0/PSP0	RD0	TTL	CMOS	Bi-directional I/O(2)
	PSP0	TTL	—	Parallel Slave Port Data Input(2)
RD1/PSP1	RD1	TTL	CMOS	Bi-directional I/O(2)
	PSP1	TTL	—	Parallel Slave Port Data Input(2)
RD2/PSP2	RD2	TTL	CMOS	Bi-directional I/O(2)
	PSP2	TTL	—	Parallel Slave Port Data Input(2)
RD3/PSP3	RD3	TTL	CMOS	Bi-directional I/O(2)
	PSP3	TTL	—	Parallel Slave Port Data Input(2)
RD4/PSP4	RD4	TTL	CMOS	Bi-directional I/O(2)
	PSP4	TTL	—	Parallel Slave Port Data Input(2)
RD5/PSP5	RD5	TTL	CMOS	Bi-directional I/O(2)
	PSP5	TTL	—	Parallel Slave Port Data Input(2)
RD6/PSP6	RD6	TTL	CMOS	Bi-directional I/O(2)
	PSP6	TTL	—	Parallel Slave Port Data Input(2)
RD7/PSP7	RD7	TTL	CMOS	Bi-directional I/O(2)
	PSP7	TTL	—	Parallel Slave Port Data Input(2)
RE0/$\overline{RD}$/AN5	RE0	ST	CMOS	Bi-directional I/O(2)
	$\overline{RD}$	TTL	—	Parallel Slave Port Control Input(2)
	AN5	AN	—	A/D Input(2)
RE1/$\overline{WR}$/AN6	RE1	ST	CMOS	Bi-directional I/O(2)
	$\overline{WR}$	TTL	—	Parallel Slave Port Control Input(2)
	AN6	AN	—	A/D Input(2)
RE2/$\overline{CS}$/AN7	RE2	ST	CMOS	Bi-directional I/O(2)
	$\overline{CS}$	TTL	—	Parallel Slave Port Data Input(2)
	AN7	AN	—	A/D Input(2)
VDD	VDD	Power	—	Power
VSS	VSS	Power	—	Ground

Legend: OD = open drain, ST = Schmitt Trigger

Note 1: Weak pull-ups. PORT B pull-ups are byte wide programmable.

2: PIC16C765 only.

3.1 Clocking Scheme/Instruction Cycle

The clock input feeds either an on-chip PLL, or directly drives (FINT). The clock output from either the PLL or direct drive (FINT) is internally divided by four to generate four non-overlapping quadrature clocks namely, Q1, Q2, Q3 and Q4. Internally, the program counter (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.

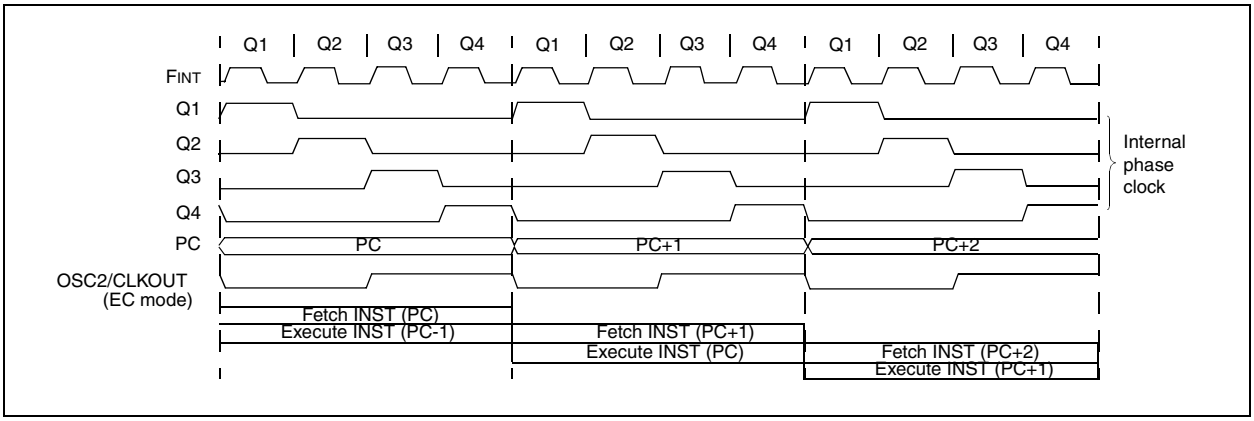
3.2 Instruction Flow/Pipelining

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 (Example 3-1).

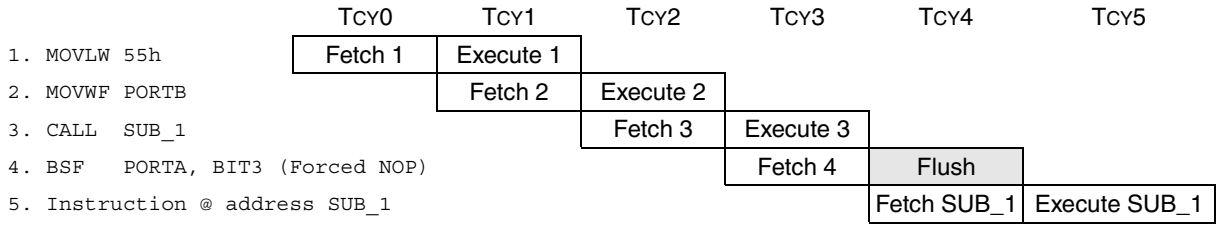
A fetch cycle begins with the program counter (PC) incrementing in Q1.

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



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

NOTES:

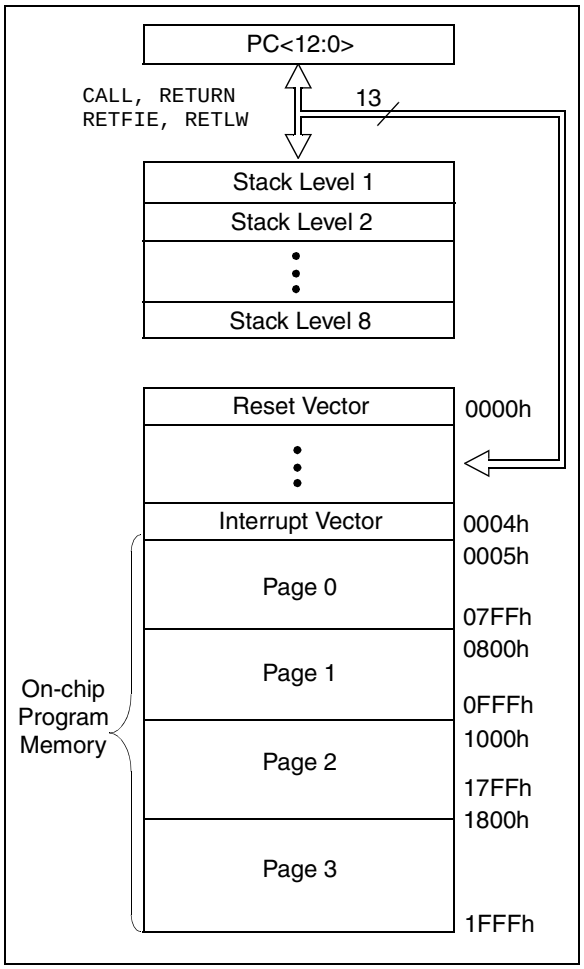
4.0 MEMORY ORGANIZATION

4.1 Program Memory Organization

The PIC16C745/765 has a 13-bit program counter capable of addressing an 8K x 14 program memory space. All devices covered by this data sheet have 8K x 14 bits of program memory. The address range is 0000h - 1FFFh for all devices.

The reset vector is at 0000h and the interrupt vector is at 0004h.

FIGURE 4-1: PIC16C745/765 PROGRAM MEMORY MAP AND STACK



4.2 Data Memory Organization

The data memory is partitioned into multiple banks which contain the General Purpose Registers (GPR) and the Special Function Registers (SFR). Bits RP1 and RP0 are the bank select bits.

RP<1:0> (STATUS<6:5>)
= 00 → Bank0
= 01 → Bank1
= 10 → Bank2
= 11 → Bank3

Each bank extends up to 7Fh (128 bytes). The lower locations of each bank are reserved for the SFRs. Above the SFRs are GPRs, implemented as static RAM.

All implemented banks contain SFRs. Some “high use” SFRs from one bank may be mirrored in another bank for code reduction and quicker access.

4.2.1 GENERAL PURPOSE REGISTER FILE

The register file can be accessed either directly or indirectly through the File Select Register (FSR) (Section 4.5).

FIGURE 4-2: DATA MEMORY MAP FOR PIC16C745/765

Bank 0	File Address	Bank 1	File Address	Bank 2	File Address	Bank 3	File Address
Indirect addr.(*)	00h	Indirect addr.(*)	80h	Indirect addr.(*)	100h	Indirect addr.(*)	180h
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h		105h		185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
PORTC	07h	TRISC	87h		107h		187h
PORTD(2)	08h	TRISD(2)	88h		108h		188h
PORTE(2)	09h	TRISE(2)	89h		109h		189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch		10Ch		18Ch
PIR2	0Dh	PIE2	8Dh		10Dh		18Dh
TMR1L	0Eh	PCON	8Eh		10Eh		18Eh
TMR1H	0Fh		8Fh		10Fh		18Fh
T1CON	10h		90h		110h	UIR	190h
TMR2	11h		91h		111h	UIE	191h
T2CON	12h	PR2	92h		112h	UEIR	192h
	13h		93h		113h	UEIE	193h
	14h		94h		114h	USTAT	194h
CCPR1L	15h		95h		115h	UCTRL	195h
CCPR1H	16h		96h		116h	UADDR	196h
CCP1CON	17h		97h		117h	USWSTAT(1)	197h
RCSTA	18h	TXSTA	98h		118h	UEP0	198h
TXREG	19h	SPBRG	99h		119h	UEP1	199h
RCREG	1Ah		9Ah		11Ah	UEP2	19Ah
CCPR2L	1Bh		9Bh		11Bh		19Bh(1)
CCPR2H	1Ch		9Ch		11Ch		19Ch(1)
CCP2CON	1Dh		9Dh		11Dh		19Dh(1)
ADRES	1Eh		9Eh		11Eh		19Eh(1)
ADCON0	1Fh	ADCON1	9Fh		11Fh		19Fh(1)
General Purpose Register 96 Bytes	20h	General Purpose Register 80 Bytes	A0h	General Purpose Register 80 Bytes	120h	USB Dual Port Memory 64 Bytes	1A0h
							1DFh
							1E0h
			EFh		16Fh		1EFh
		accesses 70h-7Fh	F0h	accesses 70h-7Fh	170h	accesses 70h-7Fh	1F0h
	7Fh		FFh		17Fh		1FFh

Unimplemented data memory locations, read as '0'.
*Not a physical register.

Note 1: Reserved registers may contain USB state information.

2: Parallel slave ports (PORTD and PORTE) not implemented on PIC16C745; always maintain these bits clear.

4.2.2 SPECIAL FUNCTION REGISTERS

The Special Function Registers are registers used by the CPU and Peripheral Modules for controlling the desired operation of the device. These registers are implemented as static RAM.

The Special Function Registers can be classified into two sets (core and peripheral). Those registers associated with the “core” functions are described in this section, and those related to the operation of the peripheral features are described in the section of that peripheral feature.

TABLE 4-1: SPECIAL FUNCTION REGISTER SUMMARY

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets (2)
Bank 0											
00h	INDF(3)	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	0000 0000
01h	TMR0	Timer0 module’s register								xxxx xxxx	uuuu uuuu
02h	PCL(3)	Program Counter's (PC) Least Significant Byte								0000 0000	0000 0000
03h	STATUS(3)	IRP(2)	RP1(2)	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	000q quuu
04h	FSR(3)	Indirect data memory address pointer								xxxx xxxx	uuuu uuuu
05h	PORTA	—	—	PORTA Data Latch when written: PORTA pins when read						--0x 0000	--0u 0000
06h	PORTB	PORTB Data Latch when written: PORTB pins when read								xxxx xxxx	uuuu uuuu
07h	PORTC	RC7	RC6	—	—	—	RC2	RC1	RC0	xx-- -xxx	uu-- -uuu
08h	PORTD(4)	PORTD Data Latch when written: PORTD pins when read								xxxx xxxx	uuuu uuuu
09h	PORTE(4)	—	—	—	—	—	RE2	RE1	RE0	---- -xxx	---- -uuu
0Ah	PCLATH(1,3)	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	---0 0000
0Bh	INTCON(3)	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(4)	ADIF	RCIF	TXIF	USBIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
0Dh	PIR2	—	—	—	–	—	—	—	CCP2IF	---- ---0	---- ---0
0Eh	TMR1L	Holding register for the Least Significant Byte of the 16-bit TMR1 register								xxxx xxxx	uuuu uuuu
0Fh	TMR1H	Holding register for the Most Significant Byte of the 16-bit TMR1 register								xxxx xxxx	uuuu uuuu
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{T1SYNC}$	TMR1CS	TMR1ON	--00 0000	--uu uuuu
11h	TMR2	Timer2 module’s register								0000 0000	0000 0000
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	-000 0000
13h	—	Unimplemented								—	—
14h	—	Unimplemented								—	—
15h	CCPR1L	Capture/Compare/PWM Register1 (LSB)								xxxx xxxx	uuuu uuuu
16h	CCPR1H	Capture/Compare/PWM Register1 (MSB)								xxxx xxxx	uuuu uuuu
17h	CCP1CON	—	—	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	--00 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
19h	TXREG	USART Transmit Data Register								0000 0000	0000 0000
1Ah	RCREG	USART Receive Data Register								0000 0000	0000 0000
1Bh	CCPR2L	Capture/Compare/PWM Register2 (LSB)								xxxx xxxx	uuuu uuuu
1Ch	CCPR2H	Capture/Compare/PWM Register2 (MSB)								xxxx xxxx	uuuu uuuu
1Dh	CCP2CON	—	—	DC2B1	DC2B1	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	--00 0000
1Eh	ADRES	A/D Result Register								xxxx xxxx	uuuu uuuu
1Fh	ADCON0	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{DONE}$	—	ADON	0000 00-0	0000 00-0

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented read as '0'. Shaded locations are unimplemented, read as ‘0’.

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC<12:8> whose contents are transferred to the upper byte of the program counter.

2: Other (non power-up) RESETS include external RESET through $\overline{MCLR}$ and Watchdog Timer Reset.

3: These registers can be addressed from any bank.

4: The Parallel Slave Port (PORTD and PORTE) is not implemented on the PIC16C745, always maintain these bits clear.

TABLE 4-1: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets (2)
Bank 1											
80h	INDF(3)	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	0000 0000
81h	OPTION	$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
82h	PCL(3)	Program Counter's (PC) Least Significant Byte								0000 0000	0000 0000
83h	STATUS(3)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	000q quuu
84h	FSR(3)	Indirect data memory address pointer								xxxx xxxx	uuuu uuuu
85h	TRISA	—	—	PORTA Data Direction Register						--11 1111	--11 1111
86h	TRISB	PORTB Data Direction Register								1111 1111	1111 1111
87h	TRISC	TRISC7	TRISC8	—	—	—	TRISC2	TRISC1	TRISC0	11-- -111	11-- -111
88h	TRISD(4)	PORTD Data Direction Register								1111 1111	1111 1111
89h	TRISE(4)	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction Bits			0000 -111	0000 -111
8Ah	PCLATH(1,3)	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	---0 0000
8Bh	INTCON(3)	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
8Ch	PIE1	PSPIE(4)	ADIE	RCIE	TXIE	USBIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
8Dh	PIE2	—	—	—	—	—	—	—	CCP2IE	---- ---0	---- ---0
8Eh	PCON	—	—	—	—	—	—	$\overline{POR}$	$\overline{BOR}$	---- --qq	---- --uu
8Fh	—	Unimplemented								—	—
90h	—	Unimplemented								—	—
91h	—	Unimplemented								—	—
92h	PR2	Timer2 Period Register								1111 1111	1111 1111
93h	—	Unimplemented								—	—
94h	—	Unimplemented								—	—
95h	—	Unimplemented								—	—
96h	—	Unimplemented								—	—
97h	—	Unimplemented								—	—
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000
9Ah	—	Unimplemented								—	—
9Bh	—	Unimplemented								—	—
9Ch	—	Unimplemented								—	—
9Dh	—	Unimplemented								—	—
9Eh	—	Unimplemented								—	—
9Fh	ADCON1	—	—	—	—	—	PCFG2	PCFG1	PCFG0	---- -000	---- -000

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

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC<12:8> whose contents are transferred to the upper byte of the program counter.

2: Other (non power-up) RESETS include external RESET through $\overline{MCLR}$ and Watchdog Timer Reset.

3: These registers can be addressed from any bank.

4: The Parallel Slave Port (PORTD and PORTE) is not implemented on the PIC16C745, always maintain these bits clear.

TABLE 4-1: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets (2)
Bank 2											
100h	INDF(3)	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	0000 0000
101h	TMR0	Timer0 module's register								xxxx xxxx	uuuu uuuu
102h	PCL(3)	Program Counter's (PC) Least Significant Byte								0000 0000	0000 0000
103h	STATUS(3)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	000q quuu
104h	FSR(3)	Indirect data memory address pointer								xxxx xxxx	uuuu uuuu
105h	—	Unimplemented								—	—
106h	PORTB	PORTB Data Latch when written: PORTB pins when read								xxxx xxxx	uuuu uuuu
107h	—	Unimplemented								—	—
108h	—	Unimplemented								—	—
109h	—	Unimplemented								—	—
10Ah	PCLATH(1,3)	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	---0 0000
10Bh	INTCON(3)	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
10Ch-11Fh	—	Unimplemented								—	—

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

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC<12:8> whose contents are transferred to the upper byte of the program counter.

2: Other (non power-up) RESETS include external RESET through $\overline{MCLR}$ and Watchdog Timer Reset.

3: These registers can be addressed from any bank.

4: The Parallel Slave Port (PORTD and PORTE) is not implemented on the PIC16C745, always maintain these bits clear.

TABLE 4-1: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets (2)
Bank 3											
180h	INDF(3)	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	0000 0000
181h	OPTION_REG	$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
182h	PCL(3)	Program Counter's (PC) Least Significant Byte								0000 0000	0000 0000
183h	STATUS(3)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	000q quuu
184h	FSR(3)	Indirect data memory address pointer								xxxx xxxx	uuuu uuuu
185h	—	Unimplemented								—	—
186h	TRISB	PORTB Data Direction Register								1111 1111	1111 1111
187h	—	Unimplemented								—	—
188h	—	Unimplemented								—	—
189h	—	Unimplemented								—	—
18Ah	PCLATH(1,3)	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	---0 0000
18Bh	INTCON(3)	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
18Ch-18Fh	—	Unimplemented								—	—
190h	UIR	—	—	STALL	UIDLE	TOK_DNE	ACTIVITY	UERR	USB_RST	--00 0000	--00 0000
191h	UIE	—	—	STALL	UIDLE	TOK_DNE	ACTIVITY	UERR	USB_RST	--00 0000	--00 0000
192h	UEIR	BTS_ERR	OWN_ERR	WRT_ERR	BTO_ERR	DFN8	CRC16	CRC5	PID_ERR	0000 0000	0000 0000
193h	UEIE	BTS_ERR	OWN_ERR	WRT_ERR	BTO_ERR	DFN8	CRC16	CRC5	PID_ERR	0000 0000	0000 0000
194h	USTAT	—	—	—	ENDP1	ENDP0	IN	—	—	---x xx--	---u uu--
195h	UCTRL	—	—	SEO	PKT_DIS	DEV_ATT	RESUME	SUSPND	—	--x0 000-	--xq qqq-
196h	UADDR	—	ADDR6	ADDR5	ADDR4	ADDR3	ADDR2	ADDR1	ADDR0	-000 0000	-000 0000
197h	USWSTAT	SWSTAT7	SWSTAT6	SWSTAT5	SWSTAT4	SWSTAT3	SWSTAT2	SWSTAT1	SWSTAT0	0000 0000	0000 0000
198h	UEP0	—	—	—	—	EP_CTL_DIS	EP_OUT_EN	EP_IN_EN	EP_STALL	---- 0000	---- 0000
199h	UEP1	—	—	—	—	EP_CTL_DIS	EP_OUT_EN	EP_IN_EN	EP_STALL	---- 0000	---- 0000
19Ah	UEP2	—	—	—	—	EP_CTL_DIS	EP_OUT_EN	EP_IN_EN	EP_STALL	---- 0000	---- 0000
19Bh-19Fh	Reserved	Reserved, do not use.								0000 0000	0000 0000

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

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC<12:8> whose contents are transferred to the upper byte of the program counter.

2: Other (non power-up) RESETS include external RESET through $\overline{MCLR}$ and Watchdog Timer Reset.

3: These registers can be addressed from any bank.

4: The Parallel Slave Port (PORTD and PORTE) is not implemented on the PIC16C745, always maintain these bits clear.

TABLE 4-2: USB DUAL PORT RAM

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets (1)
1A0h	BD0OST	UOWN UOWN	DATA0/1 DATA0/1	PID3 —	PID2 —	PID1 DTS	PID0 BSTALL	— —	— —	xxxx xxxx	uuuu uuuu
1A1h	BD0OBC	—	—	—	—	Byte Count				xxxx xxxx	uuuu uuuu
1A2h	BD0OAL	Buffer Address Low								xxxx xxxx	uuuu uuuu
1A3h	—	Reserved								—	—
1A4h	BD0IST	UOWN UOWN	DATA0/1 DATA0/1	PID3 —	PID2 —	PID1 DTS	PID0 BSTALL	— —	— —	xxxx xxxx	uuuu uuuu
1A5h	BD0IBC	—	—	—	—	Byte Count				xxxx xxxx	uuuu uuuu
1A6h	BD0IAL	Buffer Address Low								xxxx xxxx	uuuu uuuu
1A7h	—	Reserved								—	—
1A8h	BD1OST	UOWN UOWN	DATA0/1 DATA0/1	PID3 —	PID2 —	PID1 DTS	PID0 BSTALL	— —	— —	xxxx xxxx	uuuu uuuu
1A9h	BD1OBC	—	—	—	—	Byte Count				xxxx xxxx	uuuu uuuu
1AAh	BD1OAL	Buffer Address Low								xxxx xxxx	uuuu uuuu
1ABh	—	Reserved								—	—
1ACh	BD1IST	UOWN UOWN	DATA0/1 DATA0/1	PID3 —	PID2 —	PID1 DTS	PID0 BSTALL	— —	— —	xxxx xxxx	uuuu uuuu
1ADh	BD1IBC	—	—	—	—	Byte Count				xxxx xxxx	uuuu uuuu
1AEh	BD1IAL	Buffer Address Low								xxxx xxxx	uuuu uuuu
1AFh	—	Reserved								—	—
1B0h	BD2OST	UOWN UOWN	DATA0/1 DATA0/1	PID3 —	PID2 —	PID1 DTS	PID0 BSTALL	— —	— —	xxxx xxxx	uuuu uuuu
1B1h	BD2OBC	—	—	—	—	Byte Count				xxxx xxxx	uuuu uuuu
1B2h	BD2OAL	Buffer Address Low								xxxx xxxx	uuuu uuuu
1B3h	—	Reserved								—	—
1B4h	BD2IST	UOWN UOWN	DATA0/1 DATA0/1	PID3 —	PID2 —	PID1 DTS	PID0 BSTALL	— —	— —	xxxx xxxx	uuuu uuuu
1B5h	BD2IBC	—	—	—	—	Byte Count				xxxx xxxx	uuuu uuuu
1B6h	BD2IAL	Buffer Address Low								xxxx xxxx	uuuu uuuu
1B7h	—	Reserved								—	—
1B8h- 1DFh		40 byte USB Buffer								xxxx xxxx	uuuu uuuu

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

Note 1: Other (non power-up) RESETS include external RESET through $\overline{\text{MCLR}}$ and Watchdog Timer Reset.

4.2.2.1 STATUS REGISTER

The STATUS register, shown in Register 4-1, contains the arithmetic status of the ALU, the RESET status and the bank select bits for data memory.

The STATUS register can be the destination for any instruction, as with any other register. If the STATUS register is the destination for an instruction that affects the Z, DC or C bits, then the write to these three bits is disabled. These bits are set or cleared according to the device logic. Furthermore, the $\overline{TO}$ 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 `000u u1uu` (where `u` = unchanged).

It is recommended that only `BCF`, `BSF`, `SWAPF` and `MOVWF` instructions be used to alter the STATUS register. These instructions do not affect the Z, C or DC bits in the STATUS register. For other instructions which do not affect status bits, see the "Instruction Set Summary."

Note 1: The C and DC bits operate as $\overline{borrow}$ and $\overline{digit\ borrow}$ bits, respectively, in subtraction. See the `SUBLW` and `SUBWF` instructions for examples.

REGISTER 4-1: STATUS REGISTER (STATUS: 03h, 83h, 103h, 183h)

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

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

bit 7: **IRP:** Register Bank Select bit (used for indirect addressing)
`1` = Bank 2, 3 (100h - 1FFh)
`0` = Bank 0, 1 (00h - FFh)

bit 6-5: **RP<1:0>:** Register Bank Select bits (used for direct addressing)
`00` = Bank 0 (00h - 7Fh)
`01` = Bank 1 (80h - FFh)
`10` = Bank 2 (100h - 17Fh)
`11` = Bank 3 (180h - 1FFh)

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

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

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

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

bit 0: **C:** Carry/$\overline{borrow}$ bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions)(1)
`1` = A carry-out from the most significant bit of the result occurred
`0` = No carry-out from the most significant bit of the result occurred

Note 1: For $\overline{borrow}$ the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand. For rotate (`RRF`, `RLF`) instructions, this bit is loaded with either the high or low order bit of the source register.

4.2.2.2 OPTION REGISTER

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

Note:	To achieve a 1:1 prescaler assignment for the TMR0 register, assign the prescaler to the Watchdog Timer.

REGISTER 4-2: OPTION REGISTER (OPTION_REG: 81h, 181h)

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

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

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

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

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

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

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

bit 2-0: **PS<2:0>**: Prescaler Rate Select bits

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

4.2.2.3 INTCON REGISTER

The INTCON register is a readable and writable register, which contains various enable and flag bits for the TMR0 register overflow, RB Port change and external RB0/INT pin interrupts.

Note:	Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the global enable bit, GIE (INTCON<7>). User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

REGISTER 4-3: INTERRUPT CONTROL REGISTER (INTCON: 10Bh, 18Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
bit7							bit0

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

bit 7: **GIE:** Global Interrupt Enable bit
1 = Enables all un-masked interrupts
0 = Disables all interrupts

bit 6: **PEIE**: Peripheral Interrupt Enable bit
1 = Enables all un-masked peripheral interrupts
0 = Disables all peripheral interrupts

bit 5: **T0IE**: TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 interrupt
0 = Disables the TMR0 interrupt

bit 4: **INTE**: RB0/INT External Interrupt Enable bit
1 = Enables the RB0/INT external interrupt
0 = Disables the RB0/INT external interrupt

bit 3: **RBIE**: RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2: **T0IF**: TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1: **INTF**: RB0/INT External Interrupt Flag bit
1 = The RB0/INT external interrupt occurred (must be cleared in software)
0 = The RB0/INT external interrupt did not occur

bit 0: **RBIF**: RB Port Change Interrupt Flag bit
1 = At least one of the RB<7:4> pins changed state (must be cleared in software)
0 = None of the RB<7:4> pins have changed state

4.2.2.4 PIE1 REGISTER

This register contains the individual enable bits for the peripheral interrupts.

Note:	Bit PEIE (INTCON<6>) must be set to enable any peripheral interrupt.

REGISTER 4-4: PERIPHERAL INTERRUPT ENABLE1 REGISTER (PIE1: 8Ch)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIE(1)	ADIE	RCIE	TXIE	USBIE	CCP1IE	TMR2IE	TMR1IE
bit7							bit0

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

bit 7: **PSPIE(1):** Parallel Slave Port Read/Write Interrupt Enable bit
1 = Enables the PSP read/write interrupt
0 = Disables the PSP read/write interrupt

bit 6: **ADIE:** A/D Converter Interrupt Enable bit
1 = Enables the A/D interrupt
0 = Disables the A/D interrupt

bit 5: **RCIE**: USART Receive Interrupt Enable bit
1 = Enables the USART receive interrupt
0 = Disables the USART receive interrupt

bit 4: **TXIE**: USART Transmit Interrupt Enable bit
1 = Enables the USART transmit interrupt
0 = Disables the USART transmit interrupt

bit 3: **USBIE**: Universal Serial Bus Interrupt Enable bit
1 = Enables the USB interrupt
0 = Disables the USB interrupt

bit 2: **CCP1IE**: CCP1 Interrupt Enable bit
1 = Enables the CCP1 interrupt
0 = Disables the CCP1 interrupt

bit 1: **TMR2IE**: TMR2 to PR2 Match Interrupt Enable bit
1 = Enables the TMR2 to PR2 match interrupt
0 = Disables the TMR2 to PR2 match interrupt

bit 0: **TMR1IE**: TMR1 Overflow Interrupt Enable bit
1 = Enables the TMR1 overflow interrupt
0 = Disables the TMR1 overflow interrupt

Note 1: Parallel slave ports not implemented on the PIC16C745; always maintain this bit clear.

4.2.2.5 PIR1 REGISTER

This register contains the individual flag bits for the peripheral interrupts.

Note:	Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the global enable bit, GIE (INTCON<7>). User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

REGISTER 4-5: PERIPHERAL INTERRUPT REGISTER1 (PIR1: 0Ch)

R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIF(1)	ADIF	RCIF	TXIF	USBIF	CCP1IF	TMR2IF	TMR1IF
bit7							bit0

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

bit 7: **PSPIF(1):** Parallel Slave Port Read/Write Interrupt Flag bit
1 = A read or a write operation has taken place (must be cleared in software)
0 = No read or write has occurred

bit 6: **ADIF**: A/D Converter Interrupt Flag bit
1 = An A/D conversion completed (must be cleared in software)
0 = The A/D conversion is not complete

bit 5: **RCIF**: USART Receive Interrupt Flag bit
1 = The USART receive buffer is full (clear by reading RCREG)
0 = The USART receive buffer is empty

bit 4: **TXIF**: USART Transmit Interrupt Flag bit
1 = The USART transmit buffer is empty (clear by writing to TXREG)
0 = The USART transmit buffer is full

bit 3: **USBIF**: Universal Serial Bus (USB) Interrupt Flag
1 = A USB interrupt condition has occurred. The specific cause can be found by examining the contents of the UIR and UIE registers.
0 = No USB interrupt conditions that are enabled have occurred.

bit 2: **CCP1IF**: CCP1 Interrupt Flag bit

Capture Mode
1 = A TMR1 register capture occurred (must be cleared in software)
0 = No TMR1 register capture occurred

Compare Mode
1 = A TMR1 register compare match occurred (must be cleared in software)
0 = No TMR1 register compare match occurred

PWM Mode
Unused in this mode

bit 1: **TMR2IF**: TMR2 to PR2 Match Interrupt Flag bit
1 = TMR2 to PR2 match occurred (must be cleared in software)
0 = No TMR2 to PR2 match occurred

bit 0: **TMR1IF**: TMR1 Overflow Interrupt Flag bit
1 = TMR1 register overflowed (must be cleared in software)
0 = TMR1 register did not overflow

Note 1: Parallel slave ports not implemented on the PIC16C745; always maintain this bit clear.

4.2.2.6 PIE2 REGISTER

This register contains the individual enable bit for the CCP2 peripheral interrupt.

REGISTER 4-6: PERIPHERAL INTERRUPT ENABLE 2 REGISTER (PIE2: 8Dh)

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0
—	—	—	—	—	—	—	CCP2IE
bit7							bit0

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

bit 7-1: **Unimplemented:** Read as '0'

bit 0: **CCP2IE**: CCP2 Interrupt Enable bit
1 = Enables the CCP2 interrupt
0 = Disables the CCP2 interrupt

4.2.2.7 PIR2 REGISTER

This register contains the CCP2 interrupt flag bit.

Note: Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the global enable bit, GIE (INTCON<7>). User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

REGISTER 4-7: PERIPHERAL INTERRUPT REGISTER 2 (PIR2: 0Dh)

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0
—	—	—	—	—	—	—	CCP2IF
bit7							bit0

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

bit 7-1: **Unimplemented:** Read as '0'

bit 0: **CCP2IF**: CCP2 Interrupt Flag bit

Capture Mode
1 = A TMR1 register capture occurred (must be cleared in software)
0 = No TMR1 register capture occurred

Compare Mode
1 = A TMR1 register compare match occurred (must be cleared in software)
0 = No TMR1 register compare match occurred

PWM Mode
Unused

4.2.2.8 PCON REGISTER

The Power Control (PCON) register contains flag bits to allow differentiation between a Power-on Reset (POR), a Brown-out Reset (BOR), a Watchdog Reset (WDT) and an external $\overline{MCLR}$ Reset.

Note:	$\overline{BOR}$ is unknown on POR. It must be set by the user and checked on subsequent RESETS to see if $\overline{BOR}$ is clear, indicating a brown-out has occurred.

REGISTER 4-8: POWER CONTROL REGISTER REGISTER (PCON: 8Eh)

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-q
—	—	—	—	—	—	$\overline{POR}$	$\overline{BOR}$
bit7							bit0

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

bit 7-2: **Unimplemented:** Read as '0'

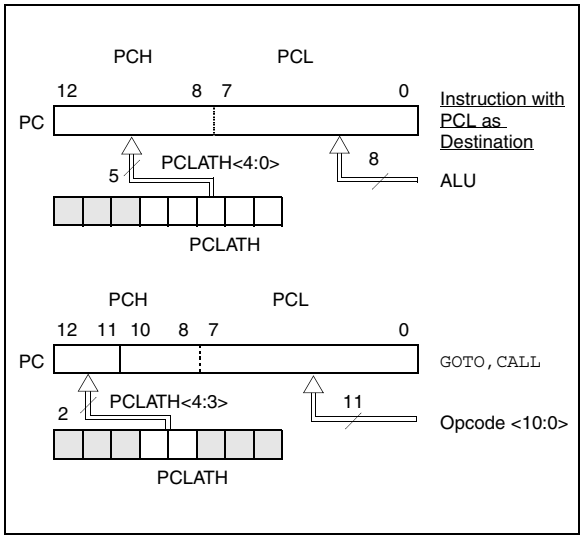
bit 1: **$\overline{POR}$**: Power-on Reset Status bit
1 = No Power-on Reset occurred
0 = A Power-on Reset occurred (must be set in software after a Power-on Reset occurs)

bit 0: **$\overline{BOR}$**: Brown-out Reset Status bit
1 = No Brown-out Reset occurred
0 = A Brown-out Reset occurred (must be set in software after a Brown-out Reset occurs)

4.3 PCL and PCLATH

The program counter (PC) is 13-bits wide. The low byte comes from the PCL register, which is a readable and writable register. The upper bits (PC<12:8>) are not readable, but are indirectly writable through the PCLATH register. On any RESET, the upper bits of the PC will be cleared. Figure 4-3 shows the two situations for the loading of the PC. The upper example in the figure shows how the PC is loaded on a write to PCL (PCLATH<4:0> → PCH). The lower example in the figure shows how the PC is loaded during a CALL or GOTO instruction (PCLATH<4:3> → PCH).

FIGURE 4-3: LOADING OF PC IN DIFFERENT SITUATIONS



4.3.1 COMPUTED GOTO

A computed GOTO is accomplished by adding an offset to the program counter (ADDWF PCL). 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 the application note *"Implementing a Table Read"* (AN556).

4.3.2 STACK

The PIC16C745/765 family has an 8-level deep x 13-bit wide hardware stack. 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 onto the stack when a CALL instruction is executed or an interrupt causes a branch. The stack is POPed in the event of a RETURN, RETLW or a RETFIE instruction execution. PCLATH is not affected by a PUSH or 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).

Note 1: There are no status bits to indicate stack overflow or stack underflow conditions.

2: There are no instructions/mnemonics called PUSH or POP. These are actions that occur from the execution of the CALL, RETURN, RETLW, and RETFIE instructions, or the vectoring to an interrupt address.

4.4 Program Memory Paging

PIC16CXX devices are capable of addressing a continuous 8K word block of program memory. The CALL and GOTO instructions provide only 11 bits of address to allow branching within any 2K program memory page. When doing a CALL or GOTO instruction, the upper 2 bits of the address are provided by PCLATH<4:3>. When doing a CALL or GOTO instruction, the user must ensure that the page select bits are programmed so that the desired program memory page is addressed. If a return from a CALL instruction (or interrupt) is executed, the entire 13-bit PC is pushed onto the stack. Therefore, manipulation of the PCLATH<4:3> bits is not required for the return instructions (which POPs the address from the stack).

Example 4-1 shows the calling of a subroutine in page 1 of the program memory. This example assumes that PCLATH is saved and restored by the interrupt service routine (if interrupts are used).

EXAMPLE 4-1: CALL OF A SUBROUTINE IN PAGE 1 FROM PAGE 0

```
        ORG  0x500
        BSF  PCLATH,3   ;Select page 1 (800h-FFFh)
        CALL SUB1_P1    ;Call subroutine in
        :               ;page 1 (800h-FFFh)
        :
        ORG  0x900      ;page 1 (800h-FFFh)
SUB1_P1
        :               ;called subroutine
        :               ;page 1 (800h-FFFh)
        :
        RETURN          ;return to Call subroutine
                        ;in page 0 (000h-7FFh)
```

4.5 Indirect Addressing, INDF and FSR Registers

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

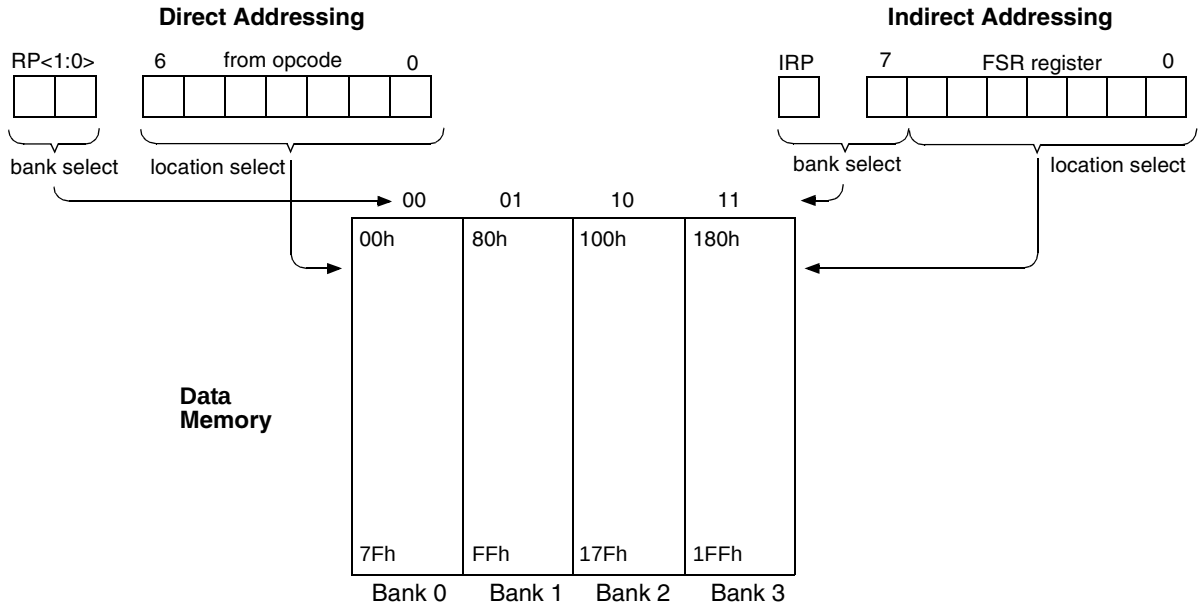
Indirect addressing is possible by using the INDF register. Any instruction using the INDF register actually accesses the register pointed to by the File Select Register, FSR. Reading the INDF register itself indirectly (FSR = '0') will read 00h. Writing to the INDF register indirectly results in a no-operation (although status bits may be affected). An effective 9-bit address is obtained by concatenating the 8-bit FSR register and the IRP bit (STATUS<7>), as shown in Figure 4-4.

A simple program to clear RAM locations 20h-2Fh using indirect addressing is shown in Example 4-2.

EXAMPLE 4-2: INDIRECT ADDRESSING

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

FIGURE 4-4: DIRECT/INDIRECT ADDRESSING



Note: For register file map detail see Figure 4-2.

5.0 I/O PORTS

Some pins for these I/O ports are multiplexed with an alternate function for the peripheral features on the device. In general, when a peripheral is enabled, that pin may not be used as a general purpose I/O pin.

5.1 PORTA and TRISA Registers

PORTA is a 6-bit latch.

The RA4/T0CKI pin is a Schmitt Trigger input and an open drain 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.

Setting a TRISA register bit puts the corresponding output driver in a hi-impedance mode. Clearing a bit in the TRISA register puts the contents of the output latch on the selected pin(s).

Reading the PORTA register reads the status of the pins, whereas writing to it will write to the port latch. All write operations are read-modify-write operations. Therefore, a write to a port implies that the port pins are read, the value is modified, and then written to the port data latch.

Pin RA4 is multiplexed with the Timer0 module clock input to become the RA4/T0CKI pin.

On the PIC16C745/765, PORTA pins are multiplexed with analog inputs and analog VREF input. The operation of each pin is selected by clearing/setting the control bits in the ADCON1 register (A/D Control Register1).

Note:	On all RESETS, pins with analog and digital functions are configured as analog inputs.

The TRISA register controls the direction of the RA pins, even when they are being used as analog inputs. The user must ensure the bits in the TRISA register are maintained set when using them as analog inputs.

EXAMPLE 5-1: INITIALIZING PORTA (PIC16C745/765)

```
BCF     STATUS, RP1     ;
BCF     STATUS, RP0     ;
CLRF    PORTA           ; Initialize PORTA by
                        ; clearing output
                        ; data latches
BSF     STATUS, RP0     ; Select Bank 1
MOVLW   0x06            ; Configure all pins
MOVWF   ADCON1          ; as digital inputs
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 RA<3:0> AND RA5 PINS

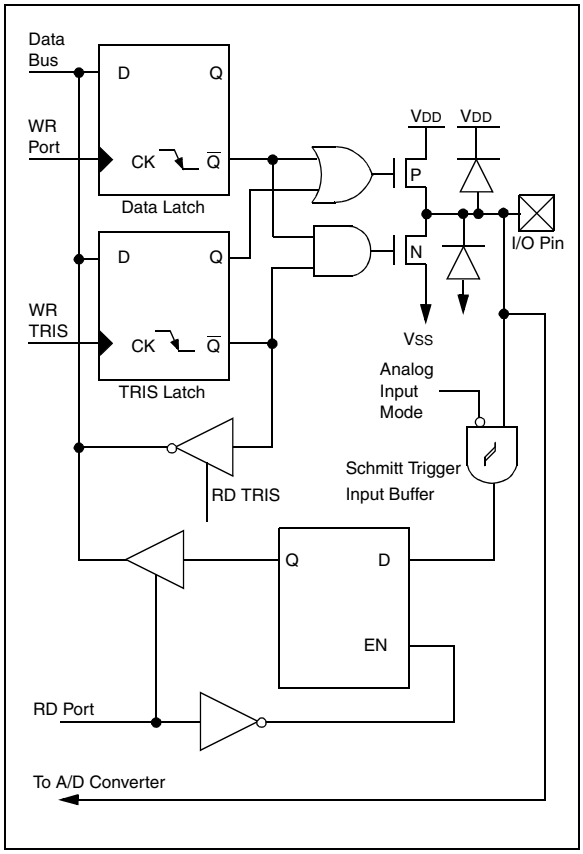


FIGURE 5-2: BLOCK DIAGRAM OF RA4/T0CKI PIN

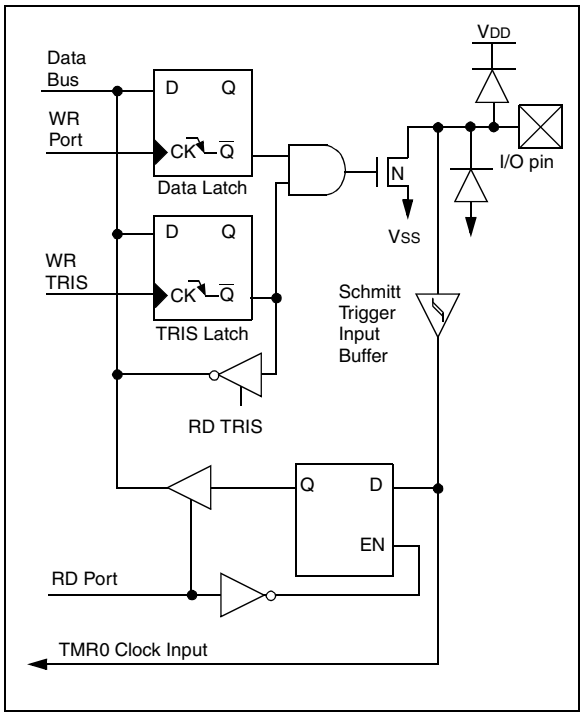


TABLE 5-1: PORTA FUNCTIONS

Name	Function	Input Type	Output Type	Description
RA0/AN0	RA0	ST	CMOS	Bi-directional I/O
	AN0	AN	—	A/D Input
RA1/AN1	RA1	ST	CMOS	Bi-directional I/O
	AN1	AN	—	A/D Input
RA2/AN2	RA2	ST	CMOS	Bi-directional I/O
	AN2	AN	—	A/D Input
RA3/AN3/VREF	RA3	ST	CMOS	Bi-directional I/O
	AN3	AN	—	A/D Input
	VREF	AN	—	A/D Positive Reference
RA4/T0CKI	RA4	ST	OD	Bi-directional I/O
	T0CKI	ST	—	Timer 0 Clock Input
RA5/AN4	RA5	ST		Bi-directional I/O
	AN4	AN	—	A/D Input

Legend: OD = open drain, ST = Schmitt Trigger

TABLE 5-2: SUMMARY OF REGISTERS ASSOCIATED WITH PORTA

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets
05h	PORTA	—	—	RA5	RA4	RA3	RA2	RA1	RA0	--0x 0000	--0u 0000
85h	TRISA	—	—	PORTA Data Direction Register						--11 1111	--11 1111
9Fh	ADCON1	—	—	—	—	—	PCFG2	PCFG1	PCFG0	---- -000	---- -000

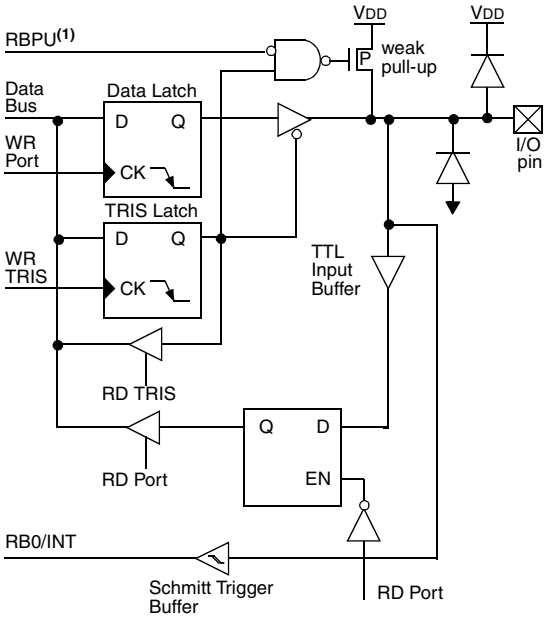
Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'. Shaded cells are not used by PORTA.

5.2 PORTB and TRISB Registers

PORTB is an 8-bit wide bi-directional port. The corresponding data direction register is TRISB. Setting a bit in the TRISB register puts the corresponding output driver in a hi-impedance input mode. Clearing a bit in the TRISB register puts the contents of the output latch on the selected pin(s).

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

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



Note 1: To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the $\overline{\text{RBPU}}$ bit (OPTION_REG<7>).

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 RB<7:4> pin configured as an output is excluded from the interrupt-on-change comparison). The input pins (of RB<7:4>) are compared with the value latched on the last read of PORTB. The "mismatch" outputs of RB<7:4> are OR'ed together to generate the RB Port Change Interrupt with flag bit RBIF (INTCON<0>).

This interrupt can wake the device from SLEEP. The user, in the interrupt service routine, can clear the interrupt in the following manner:

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

b) Clear flag bit RBIF.

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

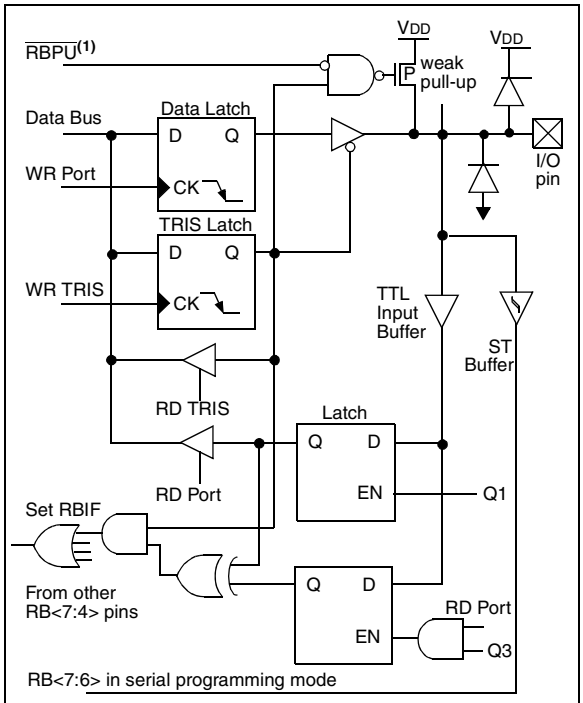
This interrupt-on-mismatch feature, together with software configureable pull-ups on these four pins, allow easy interface to a keypad and make it possible for wake-up on key depression. Refer to the Embedded Control Handbook, *"Implementing Wake-Up on Key Stroke"* (AN552).

The interrupt-on-change feature is recommended for wake-up on key depression operation and operations where PORTB is only used for the interrupt-on-change feature. Polling of PORTB is not recommended while using the interrupt-on-change feature.

RB0/INT is an external interrupt input pin and is configured using the INTEDG bit (OPTION_REG<6>).

RB0/INT is discussed in detail in Section 13.5.1.

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



Note 1: To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the $\overline{\text{RBPU}}$ bit (OPTION_REG<7>).

TABLE 5-3: PORTB FUNCTIONS

Name	Function	Input Type	Output Type	Description
RB0/INT	RB0	TTL	CMOS	Bi-directional I/O
	INT	ST	—	Interrupt
RB1	RB1	TTL	CMOS	Bi-directional I/O
RB2	RB2	TTL	CMOS	Bi-directional I/O
RB3	RB3	TTL	CMOS	Bi-directional I/O
RB4	RB4	TTL	CMOS	Bi-directional I/O with Interrupt-on-Change
RB5	RB5	TTL	CMOS	Bi-directional I/O with Interrupt-on-Change
RB6/ICSPC	RB6	TTL	CMOS	Bi-directional I/O with Interrupt-on-Change
	ICSPC	ST		In-Circuit Serial Programming Clock input
RB7/ICSPD	RB7	TTL	CMOS	Bi-directional I/O with Interrupt-on-Change
	ICSPD	ST	CMOS	In-Circuit Serial Programming Data I/O

Legend: OD = open drain, ST = Schmitt Trigger

TABLE 5-4: SUMMARY OF REGISTERS ASSOCIATED WITH PORTB

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets
06h, 106h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	uuuu uuuu
86h, 186h	TRISB	PORTB Data Direction Register								1111 1111	1111 1111
81h, 181h	OPTION_REG	$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

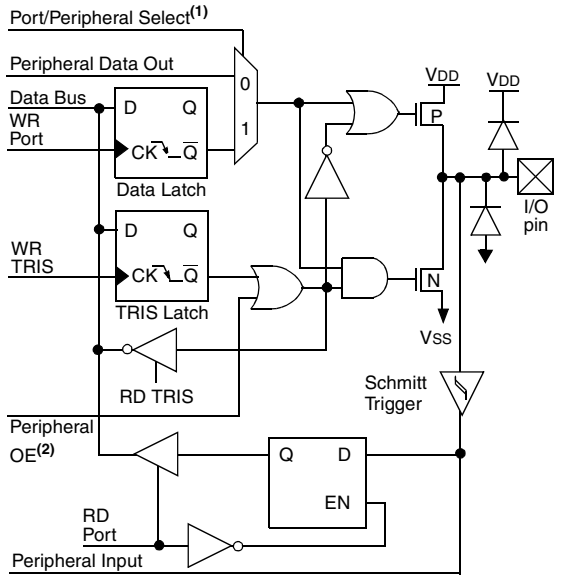
Legend: x = unknown, u = unchanged. Shaded cells are not used by PORTB.

5.3 PORTC and TRISC Registers

PORTC is a 5-bit bi-directional port. Each pin is individually configureable as an input or output through the TRISC register. PORTC is multiplexed with several peripheral functions (Table 5-5). PORTC pins have Schmitt Trigger input buffers.

When enabling peripheral functions, care should be taken in defining TRIS bits for each PORTC pin. Some peripherals override the TRIS bit to make a pin an output, while other peripherals override the TRIS bit to make a pin an input. Since the TRIS bit override is in effect while the peripheral is enabled, read-modify-write instructions (`BSF, BCF, XORWF`) with TRISC as destination should be avoided. The user should refer to the corresponding peripheral section for the correct TRIS bit settings.

FIGURE 5-5: PORTC BLOCK DIAGRAM



Note 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.

TABLE 5-5: PORTC FUNCTIONS

Name	Function	Input Type	Output Type	Description
RC0/T1OSO/T1CKI	RC0	ST	CMOS	Bi-directional I/O
	T1OSO	—	Xtal	T1 Oscillator Output
	T1CKI	ST	—	T1 Clock Input
RC1/T1OSI/CCP2	RC1	ST	CMOS	Bi-directional I/O
	T1OSI	Xtal	—	T1 Oscillator Input
	CCP2	—	—	Capture In/Compare Out/PWM Out 2
RC2/CCP1	RC2	ST	CMOS	Bi-directional I/O
	CCP1	—	—	Capture In/Compare Out/PWM Out 1
RC6/TX/CK	RC6	ST	CMOS	Bi-directional I/O
	TX	—	CMOS	USART Async Transmit
	CK	ST	CMOS	USART Master Out/Slave In Clock
RC7/RX/DT	RC7	ST	CMOS	Bi-directional I/O
	RX	ST	—	USART Async Receive
	DT	ST	CMOS	USART Data I/O

Legend: OD = open drain, ST = Schmitt Trigger

TABLE 5-6: SUMMARY OF REGISTERS ASSOCIATED WITH PORTC

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets
07h	PORTC	RC7	RC6	—	—	—	RC2	RC1	RC0	xx-- -xxx	uu-- -uuu
87h	TRISC	TRISC7	TRISC6	—	—	—	TRISC2	TRISC1	TRISC0	11-- -111	11-- -111

Legend: x = unknown, u = unchanged.

5.4 PORTD and TRISD Registers

Note: The PIC16C745 does not provide PORTD. The PORTD and TRISD registers are reserved. Always maintain these bits clear.

PORTD is an 8-bit port with Schmitt Trigger input buffers. Each pin is individually configured as an input or output.

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

FIGURE 5-6: PORTD BLOCK DIAGRAM

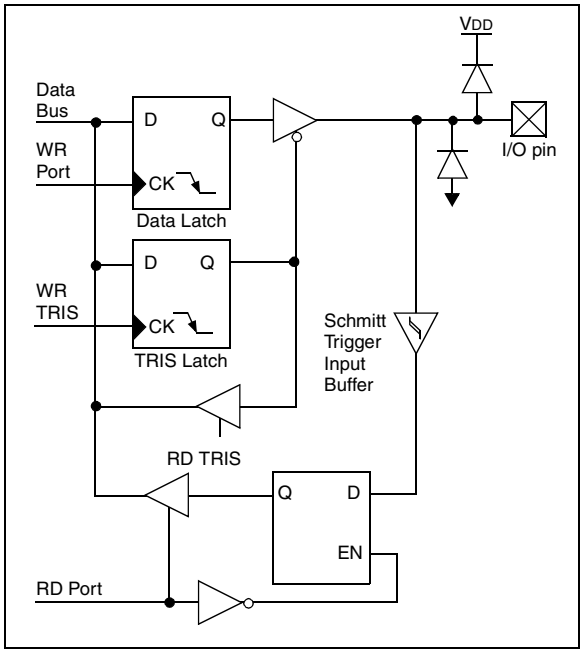


TABLE 5-7: PORTD FUNCTIONS

Name	Function	Input Type	Output Type	Description
RD0/PSP0	RD0	TTL	CMOS	Bi-directional I/O(1)
	PSP0	TTL	—	Parallel Slave Port Data Input(1)
RD1/PSP1	RD1	TTL	CMOS	Bi-directional I/O(1)
	PSP1	TTL	—	Parallel Slave Port Data Input(1)
RD2/PSP2	RD2	TTL	CMOS	Bi-directional I/O(1)
	PSP2	TTL	—	Parallel Slave Port Data Input(1)
RD3/PSP3	RD3	TTL	CMOS	Bi-directional I/O(1)
	PSP3	TTL	—	Parallel Slave Port Data Input(1)
RD4/PSP4	RD4	TTL	CMOS	Bi-directional I/O(1)
	PSP4	TTL	—	Parallel Slave Port Data Input(1)
RD5/PSP5	RD5	TTL	CMOS	Bi-directional I/O(1)
	PSP5	TTL	—	Parallel Slave Port Data Input(1)
RD6/PSP6	RD6	TTL	CMOS	Bi-directional I/O(1)
	PSP6	TTL	—	Parallel Slave Port Data Input(1)
RD7/PSP7	RD7	TTL	CMOS	Bi-directional I/O(1)
	PSP7	TTL	—	Parallel Slave Port Data Input(1)

Legend: OD = open drain, ST = Schmitt Trigger

Note 1: PIC16C765 only.

TABLE 5-8: SUMMARY OF REGISTERS ASSOCIATED WITH PORTD

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets
08h	PORTD(1)	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	xxxx xxxx	uuuu uuuu
88h	TRISD(1)	PORTD Data Direction Register								1111 1111	1111 1111

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

Note 1: PIC16C765 only.

5.5 PORTE and TRISE Registers

Note 1: The PIC16C745 does not provide PORTE. The PORTE and TRISE registers are reserved. Always maintain these bits clear.

PORTE has three pins, RE0/$\overline{RD}$/AN5, RE1/$\overline{WR}$/AN6 and RE2/$\overline{CS}$/AN7, which are individually configured as inputs or outputs. These pins have Schmitt Trigger input buffers.

I/O PORTE becomes control inputs for the microprocessor port when bit PSPMODE (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) and that register ADCON1 is configured for digital I/O. In this mode, the input buffers are TTL.

Register 5-1 shows the TRISE register, which also controls the parallel slave port operation.

PORTE pins may be multiplexed with analog inputs (PIC16C765 only). The operation of these pins is selected by control bits in the ADCON1 register. When selected as an analog input, these pins will read as '0's.

TRISE controls the direction of the RE pins, even when they are being used as analog inputs. The user must make sure to keep the pins configured as inputs when using them as analog inputs.

TRISE bits are used to control the parallel slave port.

Note: On a Power-on Reset, these pins are configured as analog inputs.

FIGURE 5-7: PORTE BLOCK DIAGRAM

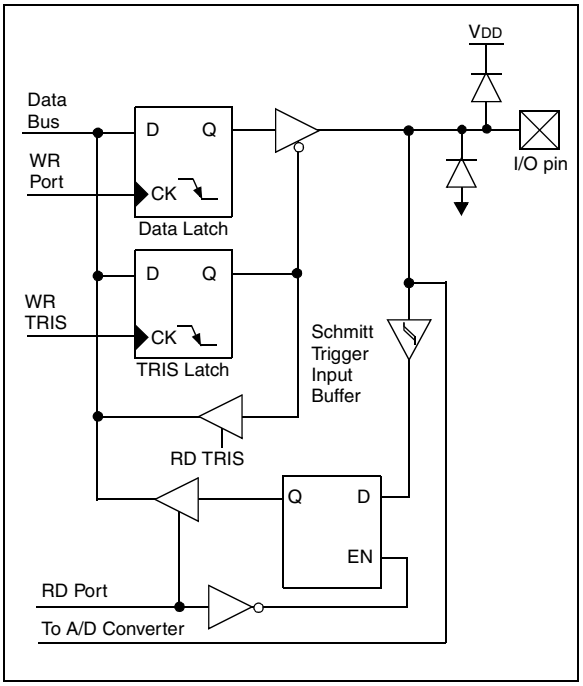


TABLE 5-9: PORTE(1) FUNCTIONS

Name	Function	Input Type	Output Type	Description
RE0/$\overline{RD}$/AN5	RE0	ST	CMOS	Bi-directional I/O(1)
	$\overline{RD}$	TTL	—	Parallel Slave Port Control Input(1)
	AN5	AN	—	A/D Input(1)
RE1/$\overline{WR}$/AN6	RE1	ST	CMOS	Bi-directional I/O(1)
	$\overline{WR}$	TTL	—	Parallel Slave Port Control Input(1)
	AN6	AN	—	A/D Input(1)
RE2/$\overline{CS}$/AN7	RE2	ST	CMOS	Bi-directional I/O(1)
	$\overline{CS}$	TTL	—	Parallel Slave Port Data Input(1)
	AN7	AN	—	A/D Input(1)

Legend: OD = open drain, ST = Schmitt Trigger

Note 1: PIC16C765 only.

REGISTER 5-1: PORTE DATA DIRECTION CONTROL REGISTER(1) (TRISE: 89h)

R-0	R-0	R/W-0	R/W-0	U-0	R/W-1	R/W-1	R/W-1
IBF	OBF	IBOV	PSPMODE	—	TRISE2	TRISE1	TRISE0
bit7							bit0

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

bit 7 : **IBF:** Input Buffer Full Status bit
1 = A word has been received and is waiting to be read by the CPU
0 = No word has been received

bit 6: **OBF**: Output Buffer Full Status bit
1 = The output buffer still holds a previously written word
0 = The output buffer has been read

bit 5: **IBOV**: Input Buffer Overflow Detect bit (in microprocessor mode)
1 = A write occurred when a previously input word has not been read (must be cleared in software)
0 = No overflow occurred

bit 4: **PSPMODE**: Parallel Slave Port Mode Select bit
1 = Parallel slave port mode
0 = General purpose I/O mode

bit 3: **Unimplemented**: Read as '0'

PORTE Data Direction Bits

bit 2: **TRISE2**: Direction Control bit for pin RE2/$\overline{CS}$/AN7
1 = Input
0 = Output

bit 1: **TRISE1**: Direction Control bit for pin RE1/$\overline{WR}$/AN6
1 = Input
0 = Output

bit 0: **TRISE0**: Direction Control bit for pin RE0/$\overline{RD}$/AN5
1 = Input
0 = Output

Note 1: PIC16C765 only.

TABLE 5-10: SUMMARY OF REGISTERS ASSOCIATED WITH PORTE

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets
09h	PORTE(1)	—	—	—	—	—	RE2	RE1	RE0	---- -xxx	---- -uuu
89h	TRISE(1)	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction Bits			0000 -111	0000 -111
9Fh	ADCON1	—	—	—	—	—	PCFG2	PCFG1	PCFG0	---- -000	---- -000

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

Note 1: PIC16C765 only.

5.6 Parallel Slave Port (PSP)

Note:	The PIC16C745 does not provide a parallel slave port. The PORTD, PORTE, TRISD and TRISE registers are reserved. Always maintain these bits clear.

PORTD operates as an 8-bit wide Parallel Slave Port (PSP), 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 $\overline{RD}$ control input pin RE0/$\overline{RD}$/AN5 and $\overline{WR}$ control input pin RE1/$\overline{WR}$/AN6.

It can directly interface to an 8-bit microprocessor data bus. The external microprocessor can read or write the PORTD latch as an 8-bit latch. Setting bit PSPMODE enables port pin RE0/$\overline{RD}$/AN5 to be the $\overline{RD}$ input, RE1/$\overline{WR}$/AN6 to be the $\overline{WR}$ input and RE2/$\overline{CS}$/AN7 to be the $\overline{CS}$ (chip select) input. For this functionality, the corresponding data direction bits of the TRISE register (TRISE<2:0>) must be configured as inputs (set) and the A/D port configuration bits PCFG<2:0> (ADCON1<2:0>) must be set, which will configure pins RE<2:0> as digital I/O.

There are actually two 8-bit latches; one for data-out (from the PIC® microcontroller) 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.

A write to the PSP occurs when both the $\overline{CS}$ and $\overline{WR}$ lines are first detected low. When either the $\overline{CS}$ or $\overline{WR}$ lines become high (level triggered), then the Input Buffer Full (IBF) status flag bit (TRISE<7>) is set on the Q4 clock cycle, following the next Q2 cycle, to signal the write is complete (Figure 5-9). The interrupt flag bit PSPIF (PIR1<7>) is also set on the same Q4 clock cycle. IBF can only be cleared by reading the PORTD input latch. The Input Buffer Overflow (IBOV) status flag bit (TRISE<5>) is set if a second write to the PSP is attempted when the previous byte has not been read out of the buffer.

A read from the PSP occurs when both the $\overline{CS}$ and $\overline{RD}$ lines are first detected low. The Output Buffer Full (OBF) status flag bit (TRISE<6>) is cleared immediately (Figure 5-10) indicating that the PORTD latch is waiting to be read by the external bus. When either the $\overline{CS}$ or $\overline{RD}$ pin becomes high (level triggered), the interrupt flag bit PSPIF is set on the Q4 clock cycle, following the next Q2 cycle, indicating that the read is complete. OBF remains low until data is written to PORTD by the user firmware.

When not in PSP mode, the IBF and OBF bits are held clear. However, if flag bit IBOV was previously set, it must be cleared in firmware.

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

FIGURE 5-8: PORTD AND PORTE BLOCK DIAGRAM (PARALLEL SLAVE PORT)

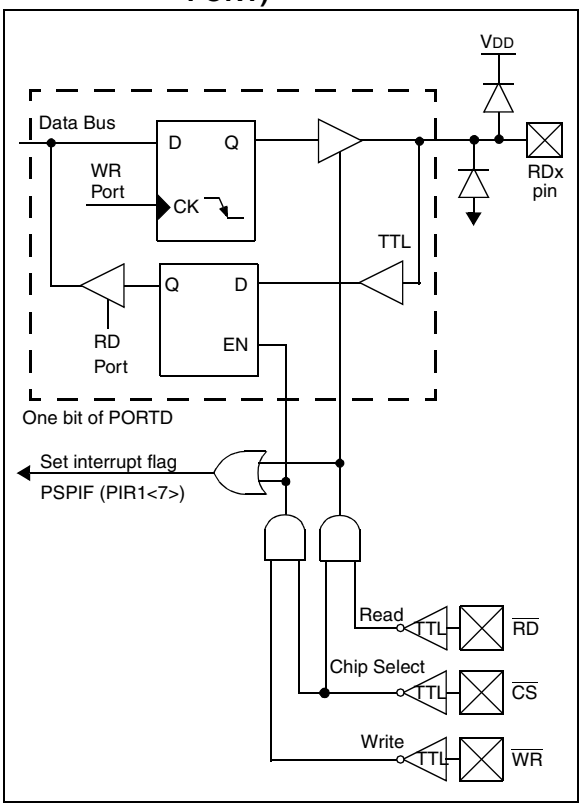


FIGURE 5-9: PARALLEL SLAVE PORT WRITE WAVEFORMS

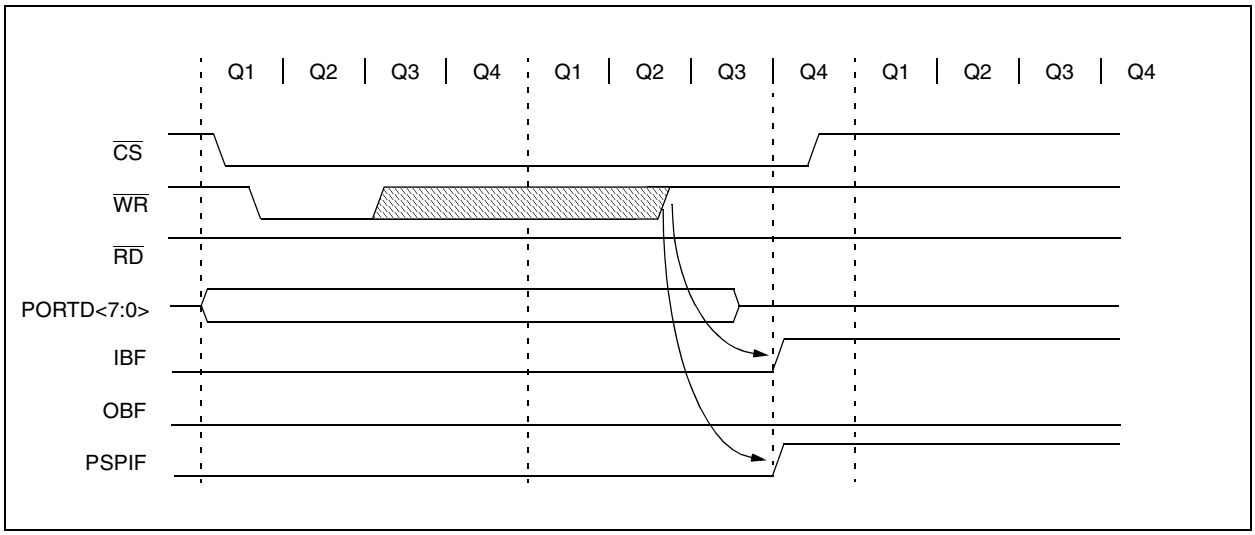


FIGURE 5-10: PARALLEL SLAVE PORT READ WAVEFORMS

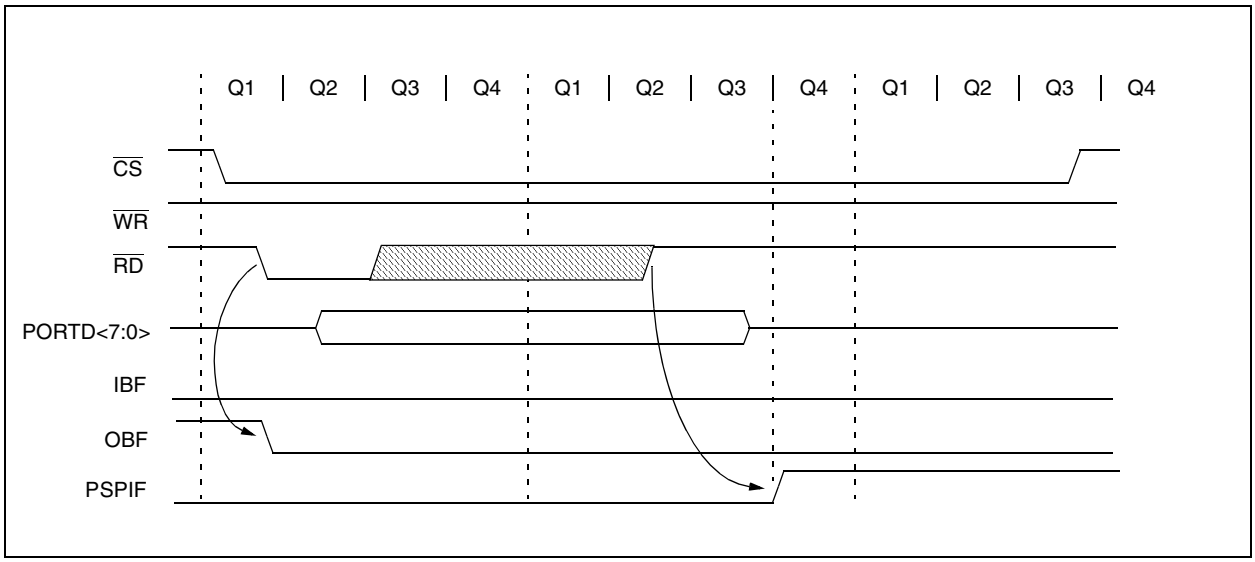


TABLE 5-11: REGISTERS ASSOCIATED WITH PARALLEL SLAVE PORT

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets
08h	PORTD(2)	Port data latch when written: Port pins when read								xxxx xxxx	uuuu uuuu
09h	PORTE(2)	—	—	—	—	—	RE2	RE1	RE0	---- -xxx	---- -uuu
89h	TRISE(2)	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction Bits			0000 -111	0000 -111
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	USBIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	USBIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
9Fh	ADCON1	—	—	—	—	—	PCFG2	PCFG1	PCFG0	---- -000	---- -000
0Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u

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

Note 1: Bits PSPIE and PSPIF are reserved on the PIC16C745. Always maintain these bits clear.

2: PIC16C765 only.

NOTES:

6.0 TIMER0 MODULE

The Timer0 module timer/counter has the following features:

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

Figure 6-1 is a block diagram of the Timer0 module and the prescaler shared with the WDT.

Additional information on the Timer0 module is available in the PIC Mid-Range MCU Family Reference Manual (DS33023).

Timer mode is selected by clearing bit T0CS (OPTION_REG<5>). In timer mode, the Timer0 module will increment every instruction cycle (without prescaler). If the TMR0 register is written, the increment is inhibited for the following two instruction cycles. The user can work around this by writing an adjusted value to the TMR0 register.

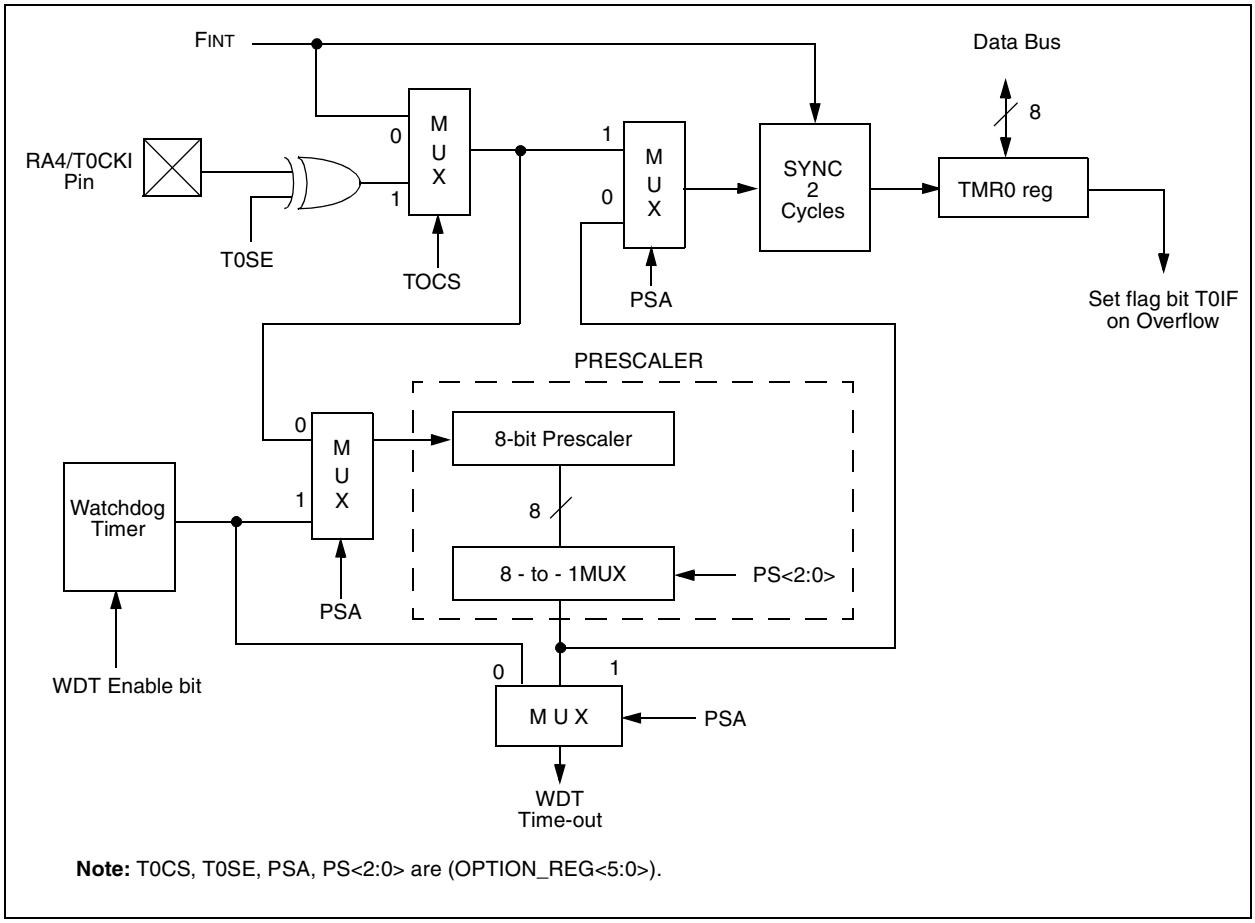
Counter mode is selected by setting bit T0CS (OPTION_REG<5>). In counter mode, Timer0 will increment either on every rising or falling edge of pin RA4/T0CKI. The incrementing edge is determined by the Timer0 Source Edge Select bit T0SE (OPTION_REG<4>). Clearing bit T0SE selects the rising edge. Restrictions on the external clock input are discussed in detail in Section 6.2.

The prescaler is mutually exclusively shared between the Timer0 module and the watchdog timer. The prescaler is not readable or writable. Section 6.3 details the operation of the prescaler.

6.1 Timer0 Interrupt

The TMR0 interrupt is generated when the TMR0 register overflows from FFh to 00h. This overflow sets bit T0IF (INTCON<2>). The interrupt can be masked by clearing bit T0IE (INTCON<5>). Bit T0IF must be cleared in software by the Timer0 module interrupt service routine before re-enabling this interrupt. The TMR0 interrupt cannot awaken the processor from SLEEP, since the timer is shut off during SLEEP.

FIGURE 6-1: BLOCK DIAGRAM OF THE TIMER0/WDT PRESCALER



Note: T0CS, T0SE, PSA, PS<2:0> are (OPTION_REG<5:0>).

6.2 Using Timer0 with an External Clock

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

6.3 Prescaler

There is only one prescaler available which is mutually exclusively shared between the Timer0 module and the watchdog timer. A prescaler assignment for the Timer0 module means that there is no prescaler for the watchdog timer, and vice-versa. This prescaler is not readable or writable (see Figure 6-1).

The PSA and PS<2:0> bits (OPTION_REG<3:0>) determine the prescaler assignment and prescale ratio.

When assigned to the Timer0 module, all instructions writing to the TMR0 register (e.g. `CLRF 1, MOVWF 1, BSF 1,x`....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.

Note:	Writing to TMR0, when the prescaler is assigned to Timer0, will clear the prescaler count, but will not change the prescaler assignment.

To avoid an unintended device RESET, the following instruction sequence (shown in Example 6-1) must be executed when changing the prescaler assignment from Timer0 to the WDT. This sequence must be followed even if the WDT is disabled.

EXAMPLE 6-1: CHANGING PRESCALER (TIMER0→WDT)

Lines 2 and 3 do NOT have to be included if the final desired prescale value is other than 1:1. If 1:1 is the final desired value, then a temporary prescale value is set in lines 2 and 3 and the final prescale value will be set in lines 10 and 11.

```
1)  BSF    STATUS, RP0    ;Bank1
2)  MOVLW  b'xx0x0xxx'    ;Select clock source and prescale value of
3)  MOVWF  OPTION_REG     ;other than 1:1
4)  BCF    STATUS, RP0    ;Bank0
5)  CLRF   TMR0           ;Clear TMR0 and prescaler
6)  BSF    STATUS, RP1    ;Bank1
7)  MOVLW  b'xxxx1xxx'    ;Select WDT, do not change prescale value
8)  MOVWF  OPTION_REG     ;
9)  CLRWDT                ;Clears WDT and prescaler
10) MOVLW  b'xxxx1xxx'    ;Select new prescale value and WDT
11) MOVWF  OPTION_REG     ;
12) BCF    STATUS, RP0    ;Bank0
```

TABLE 6-1: REGISTERS ASSOCIATED WITH TIMER0

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets
01h,101h	TMR0	Timer0 module's register								xxxx xxxx	uuuu uuuu
0Bh,8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
81h,181h	OPTION_REG	$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'. Shaded cells are not used by Timer0.

7.0 TIMER1 MODULE

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

Timer1 can operate in one of two modes:

- As a timer
- As a counter

The operating mode is determined by the clock select bit, TMR1CS (T1CON<1>).

In timer mode, Timer1 increments every instruction cycle. In counter mode, it increments on every rising edge of the external clock input.

Timer1 can be enabled/disabled by setting/clearing control bit TMR1ON (T1CON<0>).

Timer1 also has an internal “RESET input”. This RESET can be generated by either of the two CCP modules (Section 9.0). Register 7-1 shows the Timer1 control register.

When the Timer1 oscillator is enabled (T1OSCEN is set), the RC1/T1OSI/CCP2 and RC0/T1OSO/T1CKI pins become inputs. That is, the TRISC<1:0> value is ignored.

Additional information on timer modules is available in the PIC Mid-Range MCU Family Reference Manual (DS33023).

REGISTER 7-1: TIMER1 CONTROL REGISTER (T1CON: 10h)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON
bit7							bit0

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

bit 7-6: **Unimplemented:** Read as '0'

bit 5-4: **T1CKPS<1:0>**: Timer1 Input Clock Prescale Select bits
11 = 1:8 Prescale value
10 = 1:4 Prescale value
01 = 1:2 Prescale value
00 = 1:1 Prescale value

bit 3: **T1OSCEN**: Timer1 Oscillator Enable Control bit
1 = Oscillator is enabled
0 = Oscillator is shut off (The oscillator inverter is turned off to eliminate power drain)

bit 2: **$\overline{\text{T1SYNC}}$**: Timer1 External Clock Input Synchronization Control bit

TMR1CS = 1
1 = Do not synchronize external clock input
0 = Synchronize external clock input

TMR1CS = 0
This bit is ignored. Timer1 uses the internal clock when TMR1CS = 0.

bit 1: **TMR1CS**: Timer1 Clock Source Select bit
1 = External clock from pin RC0/T1OSO/T1CKI (1) or RC1/T1OSI/CCP2
0 = Internal clock (FINT)

bit 0: **TMR1ON**: Timer1 On bit
1 = Enables Timer1
0 = Stops Timer1

Note 1: On the rising edge after the first falling edge.

7.1 Timer1 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 FINT. The synchronize control bit $\overline{\text{T1SYNC}}$ (T1CON<2>) has no effect since the internal clock is always in sync.

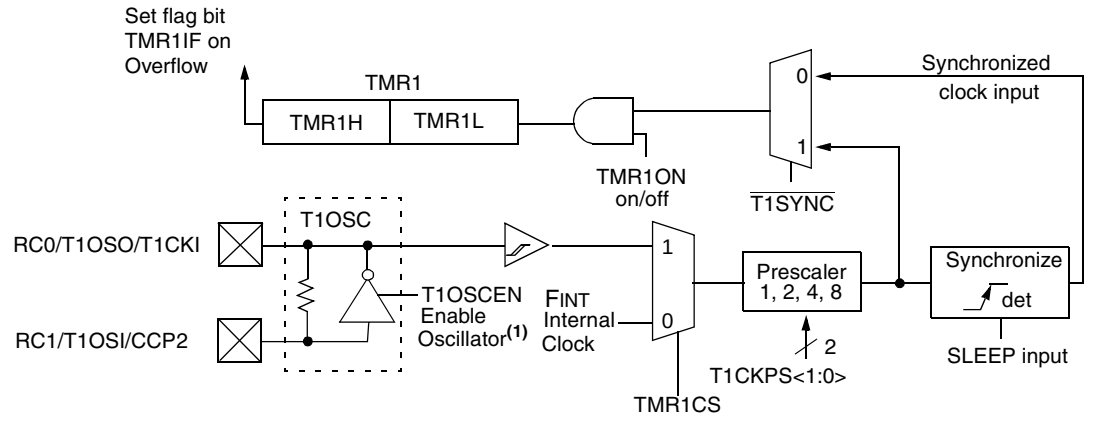
7.2 Timer1 Operation in Synchronized Counter Mode

Counter mode is selected by setting bit TMR1CS. In this mode, the timer increments on every rising edge of clock input on pin RC1/T1OSI/CCP2, when bit T1OSCEN is set, or on pin RC0/T1OSO/T1CKI, when bit T1OSCEN is cleared.

If $\overline{\text{T1SYNC}}$ 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, Timer1 will not increment even if the external clock is present, since the synchronization circuit is shut off. The prescaler however will continue to increment.

FIGURE 7-1: TIMER1 BLOCK DIAGRAM



Note 1: When the T1OSCEN bit is cleared, the inverter is turned off. This eliminates power drain.

7.3 Timer1 Operation in Asynchronous Counter Mode

If control bit $\overline{\text{T1SYNC}}$ (T1CON<2>) 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 can generate an interrupt-on-overflow, which will wake-up the processor. However, special precautions in software are needed to read/write the timer (Section 7.3.1).

In asynchronous counter mode, Timer1 can not be used as a time-base for capture or compare operations.

7.3.1 READING AND WRITING TIMER1 IN ASYNCHRONOUS COUNTER MODE

Reading TMR1H or TMR1L while the timer is running from an 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.

Reading the 16-bit value requires some care. Examples 12-2 and 12-3 in the PIC Mid-Range MCU Family Reference Manual (DS33023) show how to read and write Timer1 when it is running in asynchronous mode.

7.4 Timer1 Oscillator

A crystal oscillator circuit is built-in between pins T1OSI (input) and T1OSO (amplifier output). It is enabled by setting control bit T1OSCEN (T1CON<3>). The oscillator is a low power oscillator rated up to 200 kHz. It will continue to run during SLEEP. It is primarily intended for use with a 32 kHz crystal. Table 7-1 shows the capacitor selection for the Timer1 oscillator.

TABLE 7-1: CAPACITOR SELECTION FOR THE TIMER1 OSCILLATOR

Osc Type	Freq	C1	C2
LP	32 kHz	33 pF	33 pF
	100 kHz	15 pF	15 pF
	200 kHz	15 pF	15 pF
These values are for design guidance only.			
Crystals Tested:			
32.768 kHz	Epson C-001R32.768K-A		± 20 PPM
100 kHz	Epson C-2 100.00 KC-P		± 20 PPM
200 kHz	STD XTL 200.000 kHz		± 20 PPM

Note 1: Higher capacitance increases the stability of oscillator but also increases the start-up time.

2: Since each resonator/crystal has its own characteristics, the user should consult the resonator/crystal manufacturer for appropriate values of external components.

7.5 Resetting Timer1 using a CCP Trigger Output

If the CCP1 or CCP2 module is configured in compare mode to generate a "special event trigger" (CCP1M<3:0> = 1011), this signal will reset Timer1.

Note: The special event triggers from the CCP1 and CCP2 modules will not set interrupt flag bit TMR1IF (PIR1<0>).

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

In the event that a write to Timer1 coincides with a special event trigger from CCP1 or CCP2, the write will take precedence.

In this mode of operation, the CCPRxH:CCPRxL register pair effectively becomes the period register for Timer1.

7.6 Resetting of Timer1 Register Pair (TMR1H, TMR1L)

TMR1H and TMR1L registers are not reset to 00h on a POR or any other RESET except by the CCP1 and CCP2 special event triggers.

T1CON register is reset to 00h on a Power-on Reset or a Brown-out Reset, which shuts off the timer and leaves a 1:1 prescale. In all other RESETS, the register is unaffected.

7.7 Timer1 Prescaler

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

TABLE 7-2: 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	Value on: POR, BOR	Value on all other resets
0Bh,8Bh, 10Bh, 18Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	USBIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	USBIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
0Eh	TMR1L	Holding register for the Least Significant Byte of the 16-bit TMR1 register								xxxx xxxx	uuuu uuuu
0Fh	TMR1H	Holding register for the Most Significant Byte of the 16-bit TMR1 register								xxxx xxxx	uuuu uuuu
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	--00 0000	--uu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented read as '0'. Shaded cells are not used by the Timer1 module.
Note 1: Bits PSPIE and PSPIF are reserved on the PIC16C745; always maintain these bits clear.

8.0 TIMER2 MODULE

Timer2 is an 8-bit timer with a prescaler and a postscaler. It can be used as the PWM time-base for the PWM mode of the CCP module(s). The TMR2 register is readable and writable, and is cleared on any device RESET.

The input clock (FINT/4) has a prescale option of 1:1, 1:4 or 1:16, selected by control bits T2CKPS<1:0> (T2CON<1:0>).

The Timer2 module has an 8-bit period register PR2. Timer2 increments from 00h until it matches PR2 and then resets to 00h on the next increment cycle. PR2 is a readable and writable register. The PR2 register is initialized to FFh upon RESET.

The match output of TMR2 goes through a 4-bit postscaler (which gives a 1:1 to 1:16 scaling inclusive) to generate a TMR2 interrupt (latched in flag bit TMR2IF, (PIR1<1>)).

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

Register 8-1 shows the Timer2 control register.

Additional information on timer modules is available in the PIC Mid-Range MCU Family Reference Manual (DS33023).

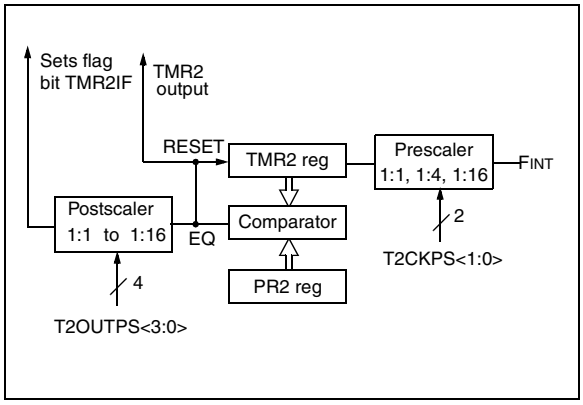
8.1 Timer2 Prescaler and Postscaler

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
- any device RESET (POR, $\overline{MCLR}$ Reset, WDT Reset or BOR)

TMR2 is not cleared when T2CON is written.

FIGURE 8-1: TIMER2 BLOCK DIAGRAM



REGISTER 8-1: TIMER2 CONTROL REGISTER (T2CON: 12h)

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
bit7							bit0

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

bit 7: **Unimplemented:** Read as '0'

bit 6-3: **TOUTPS<3:0>**: Timer2 Output Postscale Select bits
0000 = 1:1 Postscale
0001 = 1:2 Postscale
0010 = 1:3 Postscale
•
•
•
1111 = 1:16 Postscale

bit 2: **TMR2ON**: Timer2 On bit
1 = Timer2 is on
0 = Timer2 is off

bit 1-0: **T2CKPS<1:0>**: Timer2 Clock Prescale Select bits
00 = Prescaler is 1
01 = Prescaler is 4
1x = Prescaler is 16

TABLE 8-1: 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	Value on: POR, BOR	Value on all other resets
0Bh,8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	USBIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	USBIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
11h	TMR2	Timer2 module's register								0000 0000	0000 0000
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	-000 0000
92h	PR2	Timer2 Period Register								1111 1111	1111 1111

Legend: x = unknown, u = unchanged, - = unimplemented read as '0'. Shaded cells are not used by the Timer2 module.
Note 1: Bits PSPIE and PSPIF are reserved on the PIC16C745; always maintain these bits clear.

9.0 CAPTURE/COMPARE/PWM MODULES

Each Capture/Compare/PWM (CCP) module contains a 16-bit register which can operate as a:

- 16-bit capture register
- 16-bit compare register
- PWM master/slave Duty Cycle register

Both the CCP1 and CCP2 modules are identical in operation, with the exception being the operation of the special event trigger. Table 9-1 and Table 9-2 show the resources and interactions of the CCP module(s). In the following sections, the operation of a CCP module is described with respect to CCP1. CCP2 operates the same as CCP1, except where noted.

CCP1 Module:

Capture/Compare/PWM Register1 (CCPR1) is comprised of two 8-bit registers: CCPR1L (low byte) and CCPR1H (high byte). The CCP1CON register controls the operation of CCP1. The special event trigger is generated by a compare match and will reset Timer1.

CCP2 Module:

Capture/Compare/PWM Register1 (CCPR2) is comprised of two 8-bit registers: CCPR2L (low byte) and CCPR2H (high byte). The CCP2CON register controls the operation of CCP2. The special event trigger is generated by a compare match and will reset Timer1 and start an A/D conversion (if the A/D module is enabled).

Additional information on CCP modules is available in the PIC Mid-Range MCU Family Reference Manual (DS33023) and in “Using the CCP Modules” (AN594).

TABLE 9-1: CCP MODE - TIMER RESOURCES REQUIRED

CCP Mode	Timer Resource
Capture	Timer1
Compare	Timer1
PWM	Timer2

TABLE 9-2: INTERACTION OF TWO CCP MODULES

CCPx Mode	CCPy Mode	Interaction
Capture	Capture	Same TMR1 time-base.
Capture	Compare	The compare should be configured for the special event trigger, which clears TMR1.
Compare	Compare	The compare(s) should be configured for the special event trigger, which clears TMR1.
PWM	PWM	The PWMs will have the same frequency and update rate (TMR2 interrupt).
PWM	Capture	None.
PWM	Compare	None.

REGISTER 9-1: CAPTURE/COMPARE/PWM CONTROL REGISTER (CCP1CON: 17H, CCP2CON: 1Dh)

U	U	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	DCnB1	DCnB0	CCPnM3	CCPnM2	CCPnM1	CCPnM0
bit7							bit0

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

bit 7-6: **Unimplemented:** Read as '0'

bit 5-4: **DCnB<1:0>:** PWM Least Significant bits
Capture Mode: Unused
Compare Mode: Unused
PWM Mode: These bits are the two LSbs of the PWM duty cycle. The eight MSbs are found in CCPRnL.

bit 3-0: **CCPnM<3:0>:** CCPx Mode Select bits
0000 = Capture/Compare/PWM off (resets CCPn 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 (CCPnIF bit is set)
1001 = Compare mode, clear output on match (CCPnIF bit is set)
1010 = Compare mode, generate software interrupt on match (CCPnIF bit is set, CCPn pin is unaffected)
1011 = Compare mode, trigger special event (CCPnIF bit is set; CCPn resets TMR1or TMR3)
11xx = PWM mode

9.1 Capture Mode

In Capture mode, CCPR1H:CCPR1L captures the 16-bit value of the TMR1 register when an event occurs on pin RC2/CCP1. An event is defined as:

- Every falling edge
- Every rising edge
- Every 4th rising edge
- Every 16th rising edge

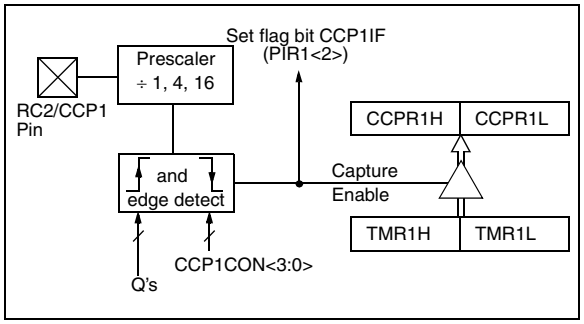
An event is selected by control bits CCP1M<3:0> (CCP1CON<3:0>). When a capture is made, the interrupt request flag bit CCP1IF (PIR1<2>) is set. The interrupt flag must be cleared in software. If another capture occurs before the value in register CCPR1 is read, the old captured value will be lost.

9.1.1 CCP PIN CONFIGURATION

In Capture mode, the RC2/CCP1 pin should be configured as an input by setting the TRISC<2> bit.

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

FIGURE 9-1: CAPTURE MODE OPERATION BLOCK DIAGRAM



9.1.2 TIMER1 MODE SELECTION

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

9.1.3 SOFTWARE INTERRUPT

When the capture mode is changed, a false capture interrupt may be generated. The user should keep bit CCP1IE (PIE1<2>) clear to avoid false interrupts and should clear the flag bit CCP1IF following any such change in operating mode.

9.1.4 CCP PRESCALER

There are four prescaler settings, specified by bits CCP1M<3:0>. Whenever the CCP module is turned off, or the CCP module is not in capture mode, the prescaler counter is cleared. 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, therefore, the first capture may be from a non-zero prescaler. Example 9-1 shows the recommended method for switching between capture prescalers. This example also clears the prescaler counter and will not generate the “false” interrupt.

EXAMPLE 9-1: CHANGING BETWEEN CAPTURE PRESCALERS

```
CLRF    CCP1CON      ;Turn CCP module off
MOVLW   NEW_CAPT_PS  ;Load the W reg with
                     ; the new precscaler
                     ; move value and CCP ON
MOVWF   CCP1CON      ;Load CCP1CON with this
                     ; value
```

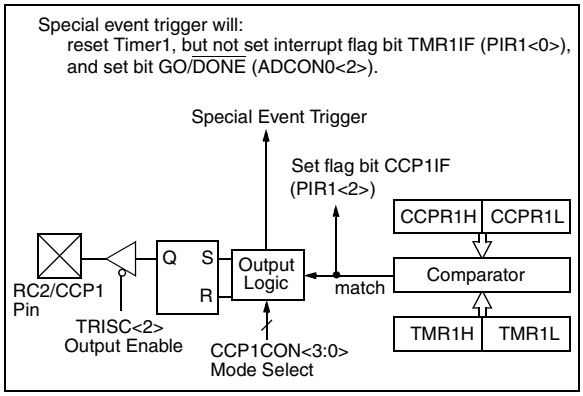
9.2 Compare Mode

In Compare mode, the 16-bit CCPR1 register value is constantly compared against the TMR1 register pair value. When a match occurs, the RC2/CCP1 pin is:

- Driven high
- Driven low
- Remains unchanged

The action on the pin is based on the value of control bits CCP1M<3:0> (CCP1CON<3:0>). At the same time, interrupt flag bit CCP1IF is set.

FIGURE 9-2: COMPARE MODE OPERATION BLOCK DIAGRAM



9.2.1 CCP PIN CONFIGURATION

The user must configure the RC2/CCP1 pin as an output by clearing 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.

9.2.2 TIMER1 MODE SELECTION

Timer1 must be running in Timer mode or Synchronized Counter mode if the CCP module is using the compare feature. In Asynchronous Counter mode, the compare operation may not work.

9.2.3 SOFTWARE INTERRUPT MODE

When Generate Software Interrupt mode is chosen, the CCP1 pin is not affected. The CCPIF bit is set causing a CCP interrupt (if enabled).

9.2.4 SPECIAL EVENT TRIGGER

In this mode, an internal hardware trigger is generated, which may be used to initiate an action.

The special event trigger output of CCP1 resets the TMR1 register pair. This allows the CCPR1 register to effectively be a 16-bit programmable period register for Timer1.

The special event trigger output of CCP2 starts an A/D conversion (if the A/D module is on) and resets the TMR1 register pair and starts an A/D conversion (if the A/D module is enabled).

Note:	The special event trigger from the CCP1and CCP2 modules will not set interrupt flag bit TMR1IF (PIR1<0>).

9.3 PWM Mode (PWM)

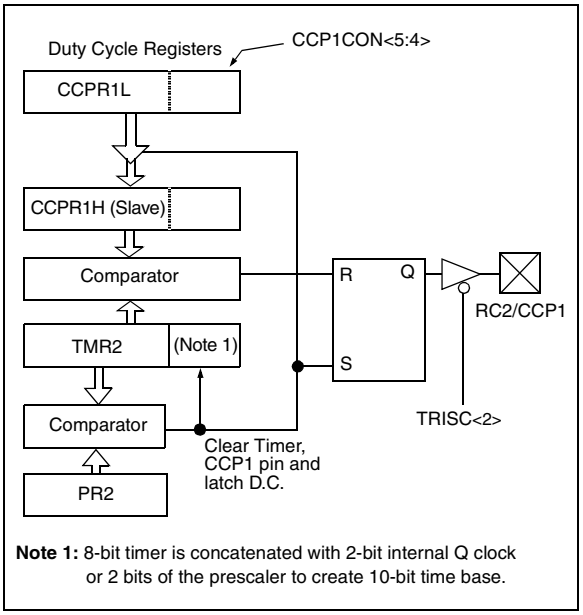
In pulse width modulation mode, the CCPx pin produces up to a 10-bit resolution PWM output. Since the CCP1 pin is multiplexed with the PORTC data latch, the TRISC<2> bit must be cleared to make the CCP1 pin an output.

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

Figure 9-3 shows a simplified block diagram of the CCP module in PWM mode.

For a step by step procedure on how to set up the CCP module for PWM operation, see Section 9.3.3.

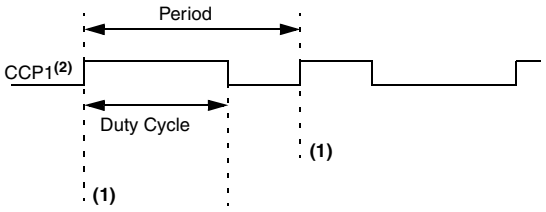
FIGURE 9-3: SIMPLIFIED PWM BLOCK DIAGRAM



Note 1: 8-bit timer is concatenated with 2-bit internal Q clock or 2 bits of the prescaler to create 10-bit time base.

A PWM output (Figure 9-4) has a time base (period) and a time that the output stays high (duty cycle). The frequency of the PWM is the inverse of the period (1/period).

FIGURE 9-4: PWM OUTPUT



Note 1: At this time, the TMR2 register is equal to the PR2 register.
2: Output signal is shown as asserted high.

9.3.1 PWM PERIOD

The PWM period is specified by writing to the PR2 register. The PWM period can be calculated using the following formula:

$$\text{PWM period} = [(PR2) + 1] \bullet 4 \bullet T_{OSC} \bullet (\text{TMR2 prescale value})$$

PWM frequency is defined as 1 / [PWM period].

When TMR2 is equal to PR2, the following three events occur on the next increment cycle:

- TMR2 is cleared
- The CCP1 pin is set (exception: if PWM duty cycle = 0%, the CCP1 pin will not be set)
- The PWM duty cycle is latched from CCPR1L into CCPR1H

Note:	The Timer2 postscaler (see Section 8.1) is not used in the determination of the PWM frequency. The postscaler could be used to have a servo update rate at a different frequency than the PWM output.

9.3.2 PWM DUTY CYCLE

The PWM duty cycle is specified by writing to the CCPR1L register and to the CCP1CON<5:4> bits. Up to 10-bit resolution is available. The CCPR1L contains the eight MSbs and the CCP1CON<5:4> contains the two LSbs. This 10-bit value is represented by CCPR1L:CCP1CON<5:4>. The following equation is used to calculate the PWM duty cycle in time:

$$\text{PWM duty cycle} = (\text{CCPR1L:CCP1CON<5:4>}) \bullet \text{Tosc} \bullet (\text{TMR2 prescale value})$$

CCPR1L and CCP1CON<5:4> can be written to at any time, but the duty cycle value is not latched into CCPR1H until after a match between PR2 and TMR2 occurs (i.e., the period is complete). In PWM mode, CCPR1H is a read-only register.

The CCPR1H register and a 2-bit internal latch are used to double buffer the PWM duty cycle. This double buffering is essential for glitchless PWM operation.

When the CCPR1H and 2-bit latch match TMR2 concatenated with an internal 2-bit Q clock or 2 bits of the TMR2 prescaler, the CCP1 pin is cleared.

Maximum PWM resolution (bits) for a given PWM frequency:

$$\text{Resolution} = \frac{\log\left(\frac{F_{INT}}{F_{PWM}}\right)}{\log(2)} \text{ bits}$$

Note:	If the PWM duty cycle value is longer than the PWM period, the CCP1 pin will not be cleared.

9.3.3 SET-UP FOR PWM OPERATION

The following steps should be taken when configuring the CCP module for PWM operation:

1. Set the PWM period by writing to the PR2 register.
2. Set the PWM duty cycle by writing to the CCPR1L register and CCP1CON<5:4> bits.
3. Make the CCP1 pin an output by clearing the TRISC<2> bit.
4. Set the TMR2 prescale value and enable Timer2 by writing to T2CON.
5. Configure the CCP1 module for PWM operation.

TABLE 9-3: REGISTERS ASSOCIATED WITH CAPTURE, COMPARE, AND TIMER1

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets
0Bh,8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	USBIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
0Dh	PIR2	—	—	—	—	—	—	—	CCP2IF	---- ---0	---- ---0
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	USBIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
8Dh	PIE2	—	—	—	—	—	—	—	CCP2IE	---- ---0	---- ---0
87h	TRISC	PORTC Data Direction Register								1111 1111	1111 1111
0Eh	TMR1L	Holding register for the Least Significant Byte of the 16-bit TMR1 register								xxxx xxxx	uuuu uuuu
0Fh	TMR1H	Holding register for the Most Significant Byte of the 16-bit TMR1 register								xxxx xxxx	uuuu uuuu
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	--00 0000	--uu uuuu
15h	CCPR1L	Capture/Compare/PWM register1 (LSB)								xxxx xxxx	uuuu uuuu
16h	CCPR1H	Capture/Compare/PWM register1 (MSB)								xxxx xxxx	uuuu uuuu
17h	CCP1CON	—	—	DCnB1	DCnB0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	--00 0000
1Bh	CCPR2L	Capture/Compare/PWM register2 (LSB)								xxxx xxxx	uuuu uuuu
1Ch	CCPR2H	Capture/Compare/PWM register2 (MSB)								xxxx xxxx	uuuu uuuu
1Dh	CCP2CON	—	—	DCnB1	DCnB0	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	--00 0000

Legend: x = unknown, u = unchanged, - = unimplemented read as '0'. Shaded cells are not used by Capture and Timer1.
Note 1: The PSP is not implemented on the PIC16C745; always maintain these bits clear.

TABLE 9-4: REGISTERS ASSOCIATED WITH PWM AND TIMER2

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets
0Bh,8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	USBIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
0Dh	PIR2	—	—	—	—	—	—	—	CCP2IF	---- ---0	---- ---0
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	USBIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
8Dh	PIE2	—	—	—	—	—	—	—	CCP2IE	---- ---0	---- ---0
87h	TRISC	PORTC Data Direction Register								1111 1111	1111 1111
11h	TMR2	Timer2 module's register								0000 0000	0000 0000
92h	PR2	Timer2 module's period register								1111 1111	1111 1111
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	-000 0000
15h	CCPR1L	Capture/Compare/PWM register1 (LSB)								xxxx xxxx	uuuu uuuu
16h	CCPR1H	Capture/Compare/PWM register1 (MSB)								xxxx xxxx	uuuu uuuu
17h	CCP1CON	—	—	DCnB1	DCnB0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	--00 0000
1Bh	CCPR2L	Capture/Compare/PWM register2 (LSB)								xxxx xxxx	uuuu uuuu
1Ch	CCPR2H	Capture/Compare/PWM register2 (MSB)								xxxx xxxx	uuuu uuuu
1Dh	CCP2CON	—	—	DCnB1	DCnB0	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	--00 0000

Legend: x = unknown, u = unchanged, - = unimplemented read as '0'. Shaded cells are not used by PWM and Timer2.
Note 1: Bits PSPIE and PSPIF are reserved on the PIC16C745; always maintain these bits clear.

10.0 UNIVERSAL SERIAL BUS

10.1 Overview

This section introduces a minimum amount of information on USB. If you already have basic knowledge of USB, you can safely skip this section. If terms like Enumeration, Endpoint, IN/OUT Transactions, Transfers and Low Speed/Full Speed are foreign to you, read on.

USB was developed to address the increased connectivity needs of PC's in the PC 2000 specification. There was a base requirement to increase the bandwidth and number of devices, which could be attached. Also desired were the ability for hot swapping, user friendly operation, robust communications and low cost. The primary promoters of USB are Intel, Compaq, Microsoft and NEC.

USB is implemented as a Tiered Star topology, with the host at the top, hubs in the middle, spreading out to the individual devices at the end. USB is limited to 127 devices on the bus, and the tree cannot be more than 6 levels deep.

USB is a host centric architecture. The host is always the master. Devices are not allowed to "speak" unless "spoken to" by the host.

Transfers take place at one of two speeds. Full Speed is 12 Mb/s and Low Speed is 1.5 Mb/s. Full Speed covers the middle ground of data intensive audio and compressed video applications, while low speed supports less data intensive applications.

10.1.1 TRANSFER PROTOCOLS

Full speed supports four transfer types: Isochronous, Bulk, Interrupt and Control. Low speed supports two transfer types: Interrupt and Control. The four transfer types are described below.

- **Isochronous Transfers**, meaning equal time, guarantee a fixed amount of data at a fixed rate. This mode trades off guaranteed data accuracy for guaranteed timeliness. Data validity is not checked because there isn't time to re-send bad packets anyway and the consequences of bad data are not catastrophic.
- **Bulk Transfers** are the converse of Isochronous. Data accuracy is guaranteed, but timeliness is not.
- **Interrupt Transfers** are designed to communicate with devices which have a moderate data rate requirement. Human Interface Devices like keyboards are but one example. For Interrupt Transfers, the key is the desire to transfer data at regular intervals. USB periodically polls these devices at a fixed rate to see if there is data to transfer.
- **Control Transfers** are used for configuration purposes.

10.1.2 FRAMES

Information communicated on the bus is grouped in a format called Frames. Each Frame is 1 ms in duration and is composed of multiple transfers. Each transfer type can be repeated more than once within a frame.

10.1.3 POWER

Power has always been a concern with any device. With USB, 5 volt power is now available directly from the bus. Devices may be self-powered or bus-powered. Self-powered devices will draw power from a wall adapter or power brick. On the other hand, bus-powered devices will draw power directly from the USB bus itself. There are limits to how much power can be drawn from the USB bus. Power is expressed in terms of "unit loads" (≤100 mA). All devices, including Hubs, are guaranteed at least 1 unit load (low power), but must negotiate with the host for up to 5 unit loads (high power). If the host determines that the bus as currently configured cannot support a device's request for more unit loads, the device will be denied the extra unit loads and must remain in a low power configuration.

10.1.4 END POINTS

At the lowest level, each device controls one or more endpoints. An endpoint can be thought of as a virtual port. Endpoints are used to communicate with a device's functions. Each endpoint is a source or sink of data. Endpoints have both an In and Out direction associated with it. Each device must implement endpoint 0 to support Control Transfers for configuration. There are a maximum of 15 endpoints available for use by each full speed device and 6 endpoints for each slow speed device. Remember that the bus is host centric, so In/Out is with respect to the host and not the device.

10.1.5 ENUMERATION

Prior to communicating on the bus, the host must see that a new device has been connected and then go through an "enumeration process". This process allows the host to ask the device to introduce itself, and negotiate performance parameters, such as power consumption, transfer protocol and polling rate. The enumeration process is initiated by the host when it detects that a new device has attached itself to the bus. This takes place completely in the background from the application process.

10.1.6 DESCRIPTORS

The USB specification requires a number of different descriptors to provide information necessary to identify a device, specify its endpoints, and each endpoint's function. The five general categories of descriptors are Device, Configuration, Interface, End Point and String.

The Device descriptor provides general information such as manufacturer, product number, serial number, USB device class the product falls under, and the number of different configurations supported. There can only be one Device descriptor for any given application.

The Configuration descriptor provides information on the power requirements of the device and how many different interfaces are supported when in this configuration. There may be more than one configuration for each device, (i.e., a high power device may also support a low power configuration).

The Interface descriptor details the number of endpoints used in this interface, as well as the class driver to use should the device support functions in more than just one device class. There can only be one Interface descriptor for each configuration.

The Endpoint descriptor details the actual registers for a given function. Information is stored about the transfer types supported, direction (In/Out), bandwidth requirements and polling interval. There may be more than one endpoint in a device, and endpoints may be shared between different interfaces.

Many of the four descriptors listed above will reference or index different String descriptors. String descriptors are used to provide vendor specific or application specific information. They may be optional and are encoded in “Unicode” format.

10.1.7 DEVICE CLASSES/CLASS DRIVERS

Operating systems provide drivers which group functions together by common device types called classes. Examples of device classes include, but are not limited to, storage, audio, communications and HID (Human Interface). Class drivers for a given application are referenced in both the Device descriptor and Interface descriptor. Most applications can find a Class Driver which supports the majority of their function/command needs. Vendors who have a requirement for specific commands which are not supported by any of the standard class drivers may provide a vendor specific “.inf” file or driver for extra support.

10.1.8 SUMMARY

While a complete USB overview is beyond the scope of this document, a few key concepts must be noted. Low speed communication is designed for devices, which in the past, used an interrupt to communicate with the host. In the USB scheme, devices do not directly interrupt the processor when they have data. Instead the host periodically polls each device to see if they have any data. This polling rate is negotiated between the device and host, giving the system a guaranteed latency.

For more details on USB, see the USB V1.1 spec, available from the USB website at www.usb.org.

10.2 Introduction

The PIC16C745/765 USB peripheral module supports Low Speed control and interrupt (IN and OUT) transfers only. The implementation supports 3 endpoint numbers (0, 1, 2) for a total of 6 endpoints.

The following terms are used in the description of the USB module:

- MCU - The core processor and corresponding firmware
- SIE - Serial Interface Engine: That part of the USB that performs functions such as CRC generation and clocking of the D+ and D- signals.
- USB - The USB module including SIE and registers
- Bit Stuffing - forces insertion of a transition on D+ and D- to maintain clock synchronization
- BD - Buffer Descriptor
- BDT - Buffer Descriptor Table
- EP - Endpoint (combination of endpoint number and direction)
- IN - Packet transfer into the host
- OUT - Packet transfer out of the host

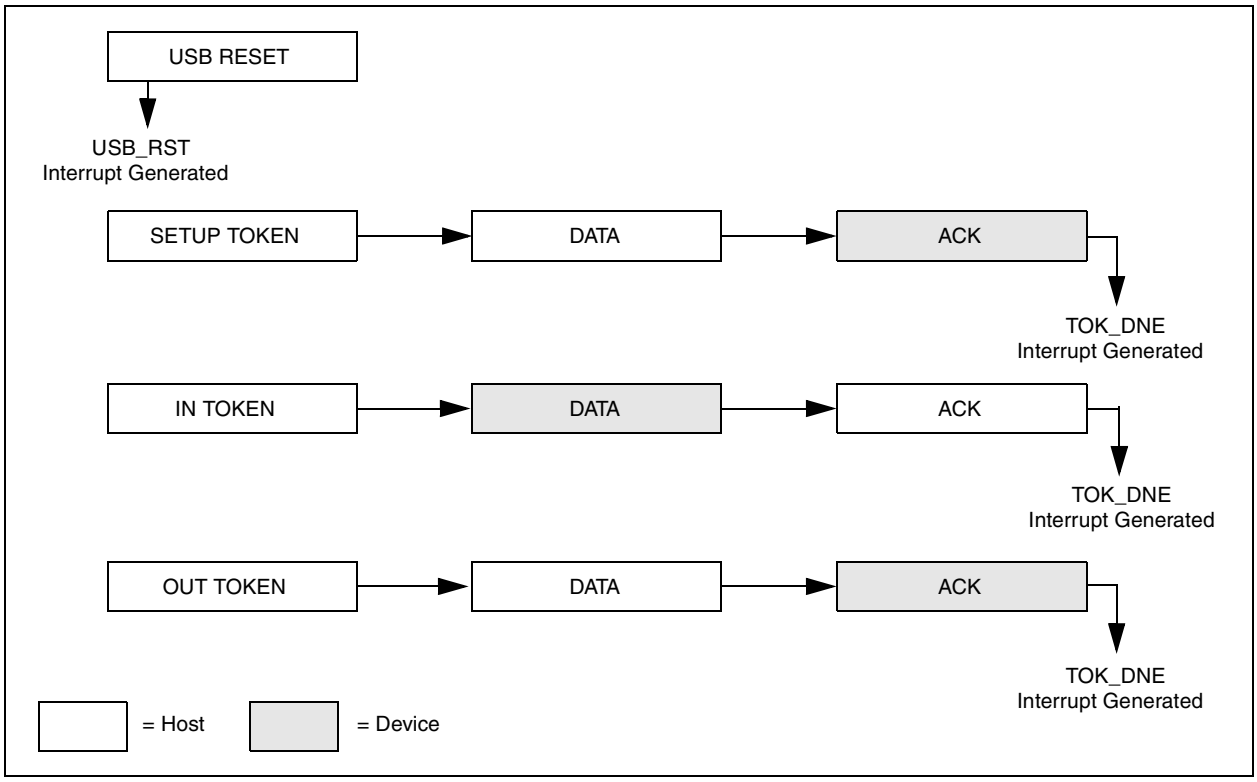
10.3 USB Transaction

When the USB transmits or receives data the SIE will first check that the corresponding endpoint and direction Buffer Description UOWN bit equals 1. The USB will move the data to or from the corresponding buffer. When the TOKEN is complete, the USB will update the BD status and change the UOWN bit to 0. The USTAT register is updated and the TOK_DNE interrupt is set. When the MCU processes the TOK_DNE interrupt it reads the USTAT register, which gives the MCU the information it needs to process the endpoint. At this point the MCU will process the data and set the corresponding UOWN bit. Figure 10-1 shows a time line of how a typical USB token would be processed.

10.4 Firmware Support

Microchip provides a comprehensive support library of standard chapter 9 USB commands. These libraries provide a software layer to insulate the application software from having to handle the complexities of the USB protocol. A simple Put/Get interface is implemented to allow most of the USB processing to take place in the background within the USB interrupt service routine. Applications are encouraged to use the provided libraries during both enumeration and configured operation.

FIGURE 10-1: USB TOKENS



10.5 USB Register Map

The USB Control Registers, Buffer Descriptors and Buffers are located in Bank 3.

10.5.1 CONTROL AND STATUS REGISTERS

The USB module is controlled by 7 registers, plus those that control each endpoint and endpoint/direction buffer.

10.5.1.1 USB Interrupt Register (UIR)

The USB Interrupt Status Register (UIR) contains flag bits for each of the interrupt sources within the USB. Each of these bits are qualified with their respective interrupt enable bits (see the Interrupt Enable Register UIE). All bits of the register are logically OR'ed together to form a single interrupt source for the microprocessor interrupt found in PIR1 (USBIF). Once an interrupt bit has been set, it must be cleared by writing a zero.

REGISTER 10-1: USB INTERRUPT FLAGS REGISTER (UIR: 190h)

U-0	U-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0
—	—	STALL	UIDLE	TOK_DNE	ACTIVITY	UERR	USB_RST
bit7							bit0

R = Readable bit
C = Clearable bit
U = Unimplemented bit, read as '0'
-n = Value at POR reset

bit 7-6: **Unimplemented:** Read as '0'

bit 5: **STALL:** A STALL handshake was sent by the SIE

bit 4: **UIDLE:** This bit is set if the USB has detected a constant idle on the USB bus signals for 3 ms. The idle timer is reset by activity on the USB bus. Once a IDLE condition has been detected, the user may wish to place the USB module in SUSPEND by setting the SUSPEND bit in the UCTRL register.

bit 3: **TOK_DNE:** This bit is set when the current token being processed is complete. The microprocessor should immediately read the USTAT register to determine the Endpoint number and direction used for this token. Clearing this bit causes the USTAT register to be cleared or the USTAT holding register to be loaded into the STAT register if another token has been processed.

bit 2: **ACTIVITY:** Activity on the D+/D- lines will cause the SIE to set this bit. Typically this bit is unmasked following detection of SLEEP. Users must enable the activity interrupt in the USB Interrupt Register (UIE: 191h) prior to entering suspend.

bit 1: **UERR:** This bit is set when any of the error conditions within the ERR_STAT register has occurred. The MCU must then read the ERR_STAT register to determine the source of the error.

bit 0: **USB_RST:** This bit is set when the USB has decoded a valid USB Reset. This will inform the MCU to write 00h into the address register and enable endpoint 0. USB_RST is set once a USB Reset has been detected for 2.5 microseconds. It will not be asserted again until the USB Reset condition has been removed, and then reasserted.

Note 1: Bits can only be modified when UCTRL.SUSPND = 0.

10.5.1.2 USB Interrupt Enable Register (UIE)

The USB Interrupt Enable Register (UIE) contains enable bits for each of the interrupt sources within the USB. Setting any of these bits will enable the respective interrupt source in the UIR register. The values in the UIE register only affect the propagation of an interrupt condition to the PIE1 register. Interrupt conditions can still be polled and serviced.

REGISTER 10-2: USB INTERRUPT ENABLE REGISTER (UIE: 191h)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	STALL	UIDLE	TOK_DNE	ACTIVITY	UERR	USB_RST
bit7							bit0

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

bit 7-6: **Unimplemented:** Read as '0'

bit 5: **STALL:** Set to enable STALL interrupts
1 = STALL interrupt enabled
0 = STALL interrupt disabled

bit 4: **UIDLE:** Set to enable IDLE interrupts
1 = IDLE interrupt enabled
0 = IDLE interrupt disabled

bit 3: **TOK_DNE:** Set to enable TOK_DNE interrupts
1 = TOK_DNE interrupt enabled
0 = TOK_DNE interrupt disabled

bit 2(1): **ACTIVITY:** Set to enable ACTIVITY interrupts
1 = ACTIVITY interrupt enabled
0 = ACTIVITY interrupt disabled

bit 1: **UERR:** Set to enable ERROR interrupts
1 = ERROR interrupt enabled
0 = ERROR interrupt disabled

bit 0: **USB_RST:** Set to enable USB_RST interrupts
1 = USB_RST interrupt enabled
0 = USB_RST interrupt disabled

Note 1: This interrupt is the only interrupt active during UCTRL.SUSPEND = 1.

10.5.1.3 USB Error Interrupt Status Register (UEIR)

The USB Error Interrupt Status Register (UEIR) contains bits for each of the error sources within the USB. Each of these bits are enabled by their respective error enable bits (UEIE). The result is OR'ed together and sent to the ERROR bit of the UIR register. Once an interrupt bit has been set it must be cleared by writing a zero to the respective interrupt bit. Each bit is set as soon as the error condition is detected. Thus, the interrupt will typically not correspond with the end of a token being processed.

REGISTER 10-3: USB ERROR INTERRUPT FLAGS STATUS REGISTER (UEIR: 192h)

R/C-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0
BTS_ERR	OWN_ERR	WRT_ERR	BTO_ERR	DFN8	CRC16	CRC5	PID_ERR
bit7							bit0

R = Readable bit
C = Clearable bit
U = Unimplemented bit, read as '0'
-n = Value at POR reset

bit 7: **BTS_ERR:** A bit stuff error has been detected

bit 6: **OWN_ERR:** This bit is set if the USB is processing a token and the OWN bit within the BDT is equal to 0 (signifying that the microprocessor owns the BDT and the SIE does not have access to the BDT). If processing an IN TOKEN this would cause a transmit data underflow condition. Processing an OUT or SETUP TOKEN would cause a receive data overflow condition.

bit 5: **WRT_ERR:** Write Error
A write by the MCU to the USB Buffer Descriptor Table or Buffer area was unsuccessful. This error occurs when the MCU attempts to write to the same location that is currently being written to by the SIE.

bit 4: **BTO_ERR:** This bit is set if a bus turnaround time-out error has occurred. This USB uses a bus turnaround timer to keep track of the amount of time elapsed between the token and data phases of a SETUP or OUT TOKEN or the data and handshake phases of a IN TOKEN. If more than 17-bit times are counted from the previous EOP before a transition from IDLE, a bus turnaround time-out error will occur.

bit 3: **DFN8:** The data field received was not 8 bits. The USB Specification 1.1 specifies that data field must be an integral number of bytes. If the data field was not an integral number of bytes this bit will be set.

bit 2: **CRC16:** The CRC16 failed

bit 1: **CRC5:** This interrupt will detect CRC5 error in the token packets generated by the host. If set the token packet was rejected due to a CRC5 error.

bit 0: **PID_ERR:** The PID check field failed

Note 1: Bits can only be modified when UCTRL.SUSPND = 0.

10.5.1.4 Error Interrupt Enable Register (UEIE)

The USB Error Interrupt Enable Register (UEIE) contains enable bits for each of the error interrupt sources within the USB. Setting any of these bits will enable the respective error interrupt source in the UEIR register.

REGISTER 10-4: USB ERROR INTERRUPT ENABLE REGISTER (UEIE: 193h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BTS_ERR	OWN_ERR	WRT_ERR	BTO_ERR	DFN8	CRC16	CRC5	PID_ERR
bit7							bit0

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

bit 7: **BTS_ERR:** Set this bit to enable BTS_ERR interrupts
1 = BTS_ERR interrupt enabled
0 = BTS_ERR interrupt disabled

bit 6: **OWN_ERR:** Set this bit to enable OWN_ERR interrupts
1 = OWN_ERR interrupt enabled
0 = OWN_ERR interrupt disabled

bit 5: **WRT_ERR:** Set this bit to enable WRT_ERR interrupts
1 = WRT_ERR interrupt enabled
0 = WRT_ERR interrupt disabled

bit 4: **BTO_ERR:** Set this bit to enable BTO_ERR interrupts
1 = BTO_ERR interrupt enabled
0 = BTO_ERR interrupt disabled

bit 3: **DFN8:** Set this bit to enable DFN8 interrupts
1 = DFN8 interrupt enabled
0 = DFN8 interrupt disabled

bit 2: **CRC16:** Set this bit to enable CRC16 interrupts
1 = CRC16 interrupt enabled
0 = CRC16 interrupt disabled

bit 1: **CRC5:** Set this bit to enable CRC5 interrupts
1 = CRC5 interrupt enabled
0 = CRC5 interrupt disabled

bit 0: **PID_ERR:** Set this bit to enable PID_ERR interrupts
1 = PID_ERR interrupt enabled
0 = PID_ERR interrupt disabled

10.5.1.5 Status Register (USTAT)

The USB Status Register reports the transaction status within the USB. When the MCU recognizes a TOK_DNE interrupt, this register should be read to determine the status of the previous endpoint communication. The data in the status register is valid when the TOK_DNE interrupt bit is asserted.

The USTAT register is actually a read window into a status FIFO maintained by the USB. When the USB uses a BD, it updates the status register. If another USB transaction is performed before the TOK_DNE interrupt is serviced the USB will store the status of the next transaction in the STAT FIFO. Thus, the STAT register is actually a four byte FIFO which allows the MCU to process one transaction while the SIE is processing the next. Clearing the TOK_DNE bit in the INT_STAT register causes the SIE to update the STAT register with the contents of the next STAT value. If the data in the STAT holding register is valid, the SIE will immediately reassert the TOK_DNE interrupt.

REGISTER 10-5: USB STATUS REGISTER (USTAT: 194h)

U-0	U-0	U-0	R-X	R-X	R-X	U-0	U-0
—	—	—	ENDP1	ENDP0	IN	—	—
bit7							bit0

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
-n = Value at POR reset
X = Don't care

bit 7-5: **Unimplemented:** Read as '0'

bit 4-3: **ENDP<1:0>:** These bits encode the endpoint address that received or transmitted the previous token. This allows the microprocessor to determine which BDT entry was updated by the last USB transaction.

bit 2: **IN:** This bit indicates the direction of the last BD that was updated
1 = The last transaction was an IN TOKEN
0 = The last transaction was an OUT or SETUP TOKEN

bit 1-0: **Unimplemented:** Read as '0'

10.5.1.6 USB Control Register (UCTRL)

The control register provides various control and configuration information for the USB.

REGISTER 10-6: USB CONTROL REGISTER (UCTRL: 195h)

U-0	U-0	R-X	R/C-0	R/W-0	R/W-0	R/W-0	U-0
—	—	SE0	PKT_DIS	DEV_ATT	RESUME	SUSPND	—
bit7							bit0

R = Readable bit
W = Writable bit
C = Clearable bit
U = Unimplemented bit, read as '0'
-n = Value at POR reset
X = Don't care

bit 7-6: **Unimplemented:** Read as '0'

bit 5: **SE0:** Live Single Ended Zero
This status bit indicates that the D+ and D- lines are both pulled to low.
1 = Single ended zero being received
0 = Single ended zero not being received

bit 4 **PKT_DIS:** The PKT_DIS bit informs the MCU that the SIE has disabled packet transmission and reception. Clearing this bit allows the SIE to continue token processing. This bit is set by the SIE when a Setup Token is received allowing software to dequeue any pending packet transactions in the BDT before resuming token processing. The PKT_DIS bit is set under certain conditions such as back to back SETUP tokens. This bit is not set on every SETUP token and can be modified only when UCTRL.SUSPND = 0.

bit 3: **DEV_ATT:** Device Attach
Enables the 3.3V output.
1 = When DEV_ATT is set, the VUSB pin will be driven with 3.3V (nominal)
0 = The VUSB pins (D+ and D-) will be in a high impedance state

bit 2: **RESUME:** Setting this bit will allow the USB to execute resume signaling. This will allow the USB to perform remote wake-up. Software must set RESUME to 1 for 10 mS then clear it to 0 to enable remote wake-up. For more information on RESUME signaling, see Section 7.1.7.5, 11.9 and 11.4.4 in the USB 1.1 specification.
1 = Perform RESUME signaling
0 = Normal operation

bit 1: **SUSPND:** Suspends USB operation and clocks and places the module in low power mode. This bit will generally be set in response to a UIDLE interrupt. It will generally be reset after an ACTIVITY interrupt. VUSB regulation will be different between suspend and non-suspend modes. The VUSB pin will still be driven, however the transceiver outputs are disabled.
1 = USB module in power conserve mode
0 = USB module normal operation

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

10.5.1.7 USB Address Register (UADDR)

The Address Register (UADDR) contains the unique USB address that the USB will decode. The register is reset to 00h after the RESET input has gone active or the USB has decoded a USB Reset signaling. That will initialize the address register to decode address 00h as required by the USB specification. The USB address must be written by the MCU during the USB SETUP phase.

REGISTER 10-7: USB ADDRESS REGISTER (UADDR: 196h)

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	ADDR6	ADDR5	ADDR4	ADDR3	ADDR2	ADDR1	ADDR0
bit7							bit0

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

bit 7: **Unimplemented:** Read as '0'

bit 6-0: **ADDR<6:0>:** This 7-bit value defines the USB address that the USB will decode

10.5.1.8 USB Software Status Register (USWSTAT)

This register is used by the USB firmware libraries for USB status.

Warning: Writing to this register may cause the SIE to drop off the Bus.

REGISTER 10-8: RESERVED SOFTWARE LIBRARY REGISTER (USWSTAT: 197H):.

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
Reserved for CH9 Firmware						Enumeration Status	

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

bit 7-2(1):**Reserved:** Read as "X'

bit 1-0: **W Enumeration Status <1:0>:** Status of USB peripheral during enumeration
00 = Powered
01 = Default
10 = Addressed
11 = Configured

Note 1: Application should not modify these bits.

10.5.1.9 Endpoint Registers

Each endpoint is controlled by an Endpoint Control Register. The PIC16C745/765 supports Buffer Descriptors (BD) for the following endpoints:

- EP0 Out
- EP0 In
- EP1 Out
- EP1 In
- EP2 Out
- EP2 In

The user will be required to disable unused Endpoints and directions using the Endpoint Control Registers.

10.5.1.10 USB Endpoint Control Register (EPCn)

The Endpoint Control Register contains the endpoint control bits for each of the 6 endpoints available on USB for a decoded address. These four bits define the control necessary for any one endpoint. Endpoint 0 (ENDP0) is associated with control pipe 0 which is required by USB for all functions (IN, OUT, and SETUP). Therefore, after a USB_RST interrupt has been received, the microprocessor should set UEP0 to contain 06h.

Note: These registers are initialized in response to a RESET from the host. The user must modify function USBReset in USB_CH9.ASM to configure the endpoints as needed for the application.

REGISTER 10-9: USB ENDPOINT CONTROL REGISTER (UEPn: 198H-19Ah)

U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	EP_CTL_DIS	EP_OUT_EN	EP_IN_EN	EP_STALL
bit7							bit0

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

bit 7-4: **Unimplemented:** Read as '0'

bit 3-1: **EP_CTL_DIS, EP_OUT_EN, EP_IN_EN:** These three bits define if an endpoint is enabled and the direction of the endpoint. The endpoint enable/direction control is defined as follows:

EP_CTL_DIS	EP_OUT_EN	EP_IN_EN	Endpoint Enable/Direction Control
X	0	0	Disable Endpoint
X	0	1	Enable Endpoint for IN tokens only
X	1	0	Enable Endpoint for OUT tokens only
1	1	1	Enable Endpoint for IN and OUT tokens
0	1	1	Enable Endpoint for IN, OUT, and SETUP tokens

bit 0: **EP_STALL:** When this bit is set it indicates that the endpoint is stalled. This bit has priority over all other control bits in the Endpoint Enable register, but is only valid if EP_IN_EN=1 or EP_OUT_EN=1. Any access to this endpoint will cause the USB to return a STALL handshake. The EP_STALL bit can be set or cleared by the SIE. Refer to the USB 1.1 Specification, Sections 4.4.4 and 8.5.2 for more details on the STALL protocol.

10.6 Buffer Descriptor Table (BDT)

To efficiently manage USB endpoint communications the USB implements a Buffer Descriptor Table (BDT) in register space. Every endpoint requires a 4 byte Buffer Descriptor (BD) entry. Because the buffers are shared between the MCU and the USB, a simple semaphore mechanism is used to distinguish which is allowed to update the BD and buffers in system memory. The UOWN bit is cleared when the BD entry is "owned" by the MCU. When the UOWN bit is set to 1, the BD entry and the buffer in system memory is owned by the USB. The MCU should not modify the BD or its corresponding data buffer.

The Buffer Descriptors provide endpoint buffer control information for the USB and MCU. The Buffer Descriptors have different meaning based on the value of the UOWN bit.

The USB Controller uses the data stored in the BDs when UOWN = 1 to determine:

- Data0 or Data1 PID
- Data toggle synchronization enable
- Number of bytes to be transmitted or received
- Starting location of the buffer

The MCU uses the data stored in the BDs when UOWN = 0 to determine:

- Data0 or Data1 PID
- The received TOKEN PID
- Number of bytes transmitted or received

Each endpoint has a 4 byte Buffer Descriptor and points to a data buffer in the USB dual port register space. Control of the BD and buffer would typically be handled in the following fashion:

- The MCU verifies UOWN = 0, sets the BDndAL to point to the start of a buffer, if necessary fills the buffer, then sets the BDndST byte to the desired value with UOWN = 1.
- When the host commands an in or out transaction, the Serial Interface Engine (SIE) performs the following:
  - Get the buffer address
  - Read or write the buffer
  - Update the USTAT register
  - Update the buffer descriptors with the packet ID (PID) value
  - Set the data 0/1 bit
  - Update the byte count
  - Clear the UOWN bit
- The MCU is interrupted and reads the USTAT, translates that value to a BD, where the UOWN, PID, Data 0/1, and byte count values are checked.

Warning:	The bit entries should be written as a whole word instead of using BSF, BCF to affect individual bits. This is because of the dual meaning of the bits. Bit sets and clears may leave other bits set incorrectly and present incorrect data to the SIE.

REGISTER 10-10: BUFFER DESCRIPTOR STATUS REGISTER. BITS WRITTEN BY THE MCU (BDndST: 1A0h, 1A4h, 1A8h, 1ACh, 1B0h, 1B4h)

W-X	W-X	U-X	U-X	W-X	W-X	U-X	U-X
UOWN	DATA0/1	—	—	DTS	BSTALL	—	—
bit7							bit0

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
-n = Value at POR reset
X = Don't care

bit 7: **UOWN:** USB Own
This UOWN bit determines who currently owns the buffer. The SIE writes a 0 to this bit when it has completed a token. This byte of the BD should always be the last byte the MCU updates when it initializes a BD. Once the BD has been assigned to the USB, the MCU should not change it in any way.
1 = USB has exclusive access to the BD. The MCU should not modify the BD or buffer.
0 = The MCU has exclusive access to the BD. The USB ignores all other fields in the BD.

bit 6: **DATA0/1:** This bit defines the type of data toggle packet that was transmitted or received
1 = Data 1 packet
0 = Data 0 packet

bit 5-4: **Reserved:** Read as 'X'

bit 3: **DTS:** Setting this bit will enable the USB to perform Data Toggle Synchronization. If a packet arrives with an incorrect DTS, it will be ignored and the buffer will remain unchanged.
1 = Data Toggle Synchronization is performed
0 = No Data Toggle Synchronization is performed

bit 2: **BSTALL:** Buffer Stall
Setting this bit will cause the USB to issue a STALL handshake if a token is received by the SIE that would use the BD in this location. The BD is not consumed by the SIE (the own bit remains and the rest of the BD are unchanged) when a BSTALL bit is set.

bit 1-0: **Reserved:** Read as 'X'

Note: Recommend that users not use BSF, BCF due to the dual functionality of this register.

REGISTER 10-11: BUFFER DESCRIPTOR STATUS. BITS READ BY THE MCU (BDndST: 1A0h, 1A4h, 1A8h, 1ACh, 1B0h, 1B4h)

R/W-0	R/W-X	R/W-X	R/W-X	R/W-X	R/W-X	U-X	U-X
UOWN	DATA0/1	PID3	PID2	PID1	PID0	—	—
bit7							bit0

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
-n = Value at POR reset
X = Don't care

bit 7: **UOWN:** USB Own
This UOWN bit determines who currently owns the buffer. The SIE writes a 0 to this bit when it has completed a token. This byte of the BD should always be the last byte the MCU updates when it initializes a BD. Once the BD has been assigned to the USB, the MCU should not change it in any way.
1 = USB has exclusive access to the BD. The MCU should not modify the BD or buffer.
0 = The MCU has exclusive access to the BD. The USB ignores all other fields in the BD.

bit 6: **DATA0/1:** This bit defines the type of data toggle packet that was transmitted or received
1 = Data 1 packet
0 = Data 0 packet

bit 5-2: **PID<3:0>:** Packet Identifier
The received token PID value.

bit 1-0: **Reserved:** Read as 'X'

Note: Recommend that users not use BSF, BCF due to the dual functionality of this register.

REGISTER 10-12: BUFFER DESCRIPTOR BYTE COUNT (BDndBC: 1A1h, 1A5h, 1A9h, 1ADh, 1B1h, 1B5h)

U-X	U-X	U-X	U-X	R/W-X	R/W-X	R/W-X	R/W-X
—	—	—	—	BC3	BC2	BC1	BC0
bit7							bit0

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
-n = Value at POR reset
X = Don't care

bit 7-4: **Reserved:** Read as 'X'

bit 3-0: **BC<3:0>:** The Byte Count bits represent the number of bytes that will be transmitted for an IN TOKEN or received during an OUT TOKEN. Valid byte counts are 0 - 8. The SIE will change this field upon the completion of an OUT or SETUP token with the actual byte count of the data received.

REGISTER 10-13: BUFFER DESCRIPTOR ADDRESS LOW (BDndAL: 1A2h, 1A6h, 1AAh, 1AEh, 1B2h, 1B6h)

R/W-X	R/W-X	R/W-X	R/W-X	R/W-X	R/W-X	R/W-X	R/W-X
BA7	BA6	BA5	BA4	BA3	BA2	BA1	BA0
bit7							bit0

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
-n = Value at POR reset
X = Don't care

bit 7-0: **BA<7:0>:** Buffer Address
The base address of the buffer controlled by this endpoint. The upper order bit address (BA8) of the 9-bit address is assumed to be 1h. This value must point to a location within the dual port memory space, Bank 3 (1B8h - 1DFh).

Note 1: This register should always contain a value between B8h-DFh.

10.6.1 ENDPOINT BUFFERS

Endpoint buffers are located in the Dual Port RAM area. The starting location of an endpoint buffer is determined by the Buffer Descriptor.

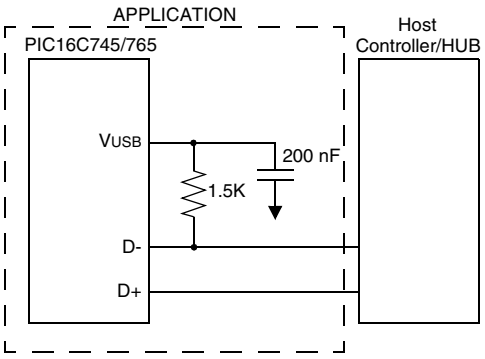
10.7 Transceiver

An on-chip integrated transceiver is included to drive the D+/D- physical layer of the USB.

10.7.1 REGULATOR

A 3.3V regulator provides the D+/D- drives with power, as well as an external pin. This pin is intended to be used to power a 1.5kΩ ±5% pull-up resistor on the D- line to signal a low speed device, as specified by the USB 1.1 Specification. A ±20% 200nF capacitor is required on VUSB for regulator stability.

FIGURE 10-2: EXTERNAL CIRCUITRY



Note: The PIC16C745/765 requires an external resistor and capacitor to communicate with a host over USB.

10.7.1.1 VUSB Output

The VUSB provides a 3.3V nominal output. This drive current is sufficient for a pull-up only.

10.8 USB Software Libraries

Microchip Technology provides a comprehensive set of Chapter 9 Standard requests functions to aid developers in implementing their designs. See Microchip Technology's website for the latest version of the software libraries.

TABLE 10-1: USB PORT FUNCTIONS

Name	Function	Input Type	Output Type	Description
VUSB	VUSB	—	Power	Regulator Output Voltage
D-	D-	USB	USB	USB Differential Bus
D+	D+	USB	USB	USB Differential Bus

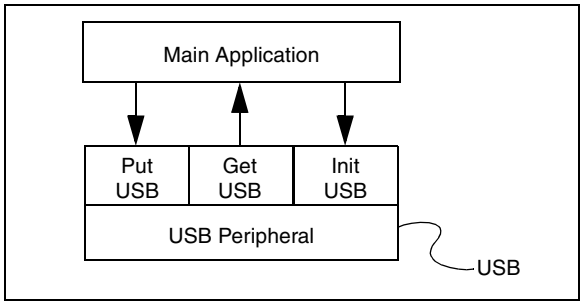
Legend: OD = open drain, ST = Schmitt Trigger

10.9 USB Firmware Users Guide

10.9.1 INTRODUCING THE USB SOFTWARE INTERFACE

Microchip provides a layer of software that handles the lowest level interface so your application won't have to. This provides a simple Put/Get interface for communication. Most of the USB processing takes place in the background through the Interrupt Service Routine. From the application viewpoint, the enumeration process and data communication takes place without further interaction. However, substantial setup is required in the form of generating appropriate descriptors.

FIGURE 10-3: USB SOFTWARE INTERFACE



10.9.2 INTEGRATING USB INTO YOUR APPLICATION

The latest version of the USB interface software is available on Microchip's website (see http://www.microchip.com/).

The interface to the application is packaged in 3 functions: **InitUSB, PutUSB** and **GetUSB**. **InitUSB** initializes the USB peripheral, allowing the host to enumerate the device. Then, for normal data communications, function **PutUSB** sends data to the host and **GetUSB** receives data from the host.

However, there's a fair amount of setup work that must be completed. USB depends heavily on the descriptors. These are the software parameters that are communicated to the host to let it know what the device is, and how to communicate with it. See USB V1.1 spec section 9.5 for more details.

Also, code must be added to give meaning to the **SetConfiguration** command. The Chapter 9 commands call **SetConfiguration** when it receives the command. Both the descriptors and **SetConfiguration** are in **DESCRIPT.ASM**.

InitUSB enables the USB interrupt so enumeration can begin. The actual enumeration process occurs in the background, driven by the host and the Interrupt Service Routine. Macro **ConfiguredUSB** waits until the device is in the CONFIGURED state. The time required to enumerate is completely dependent on the host and bus loading.

10.9.3 INTERRUPT STRUCTURE CONCERNS

10.9.3.1 Processor Resources

Most of the USB processing occurs via the interrupt and thus is invisible to application. However, it still consumes processor resources. These include ROM, RAM, Common RAM and Stack Levels. This section attempts to quantify the impact on each of these resources, and shows ways to avoid conflicts.

If you write your own Interrupt Service Routine: W, Status, FSR and PCLATH may be corrupted by servicing the USB interrupt and must be saved.

USB_MAIN.ASM provides a skeleton ISR which does this for you, and includes tests for each of the possible interrupt bits. This provides a good starting point if you haven't already written your own.

10.9.3.2 Stack Levels

The hardware stack on the PIC® MCU is only 8 levels deep. So the worst case call between the application and ISR can only be 8 levels. The enumeration process requires 4 levels, so it's best if the main application holds off on any processing until enumeration is complete. **ConfiguredUSB** is a macro that waits until the enumeration process is complete for exactly this purpose, by testing the lower two bits of USWSTAT (`0x197`).

10.9.3.3 ROM

The code required to support the USB interrupt, including the chapter 9 interface calls, but not including the descriptor tables, is about 1kW. The descriptor and string descriptor tables can each take up to an additional 256W. The location of these parts is not restricted.

10.9.3.4 RAM

With the exception of Common RAM discussed below, servicing the USB interrupt requires ~40 bytes of RAM in Bank 2. That leaves all the General Purpose RAM in banks zero and one, plus half of bank two, available for your application to use.

10.9.3.5 Common RAM Usage

The PIC16C745/765 has 16 bytes of common RAM. These are the last 16 addresses in each bank and all refer to the same 16 bytes of memory, without regard to which register bank is currently addressed by the RP0, RP1 and IRP bits.

These are particularly useful when responding to interrupts. When an interrupt occurs, the ISR doesn't immediately know which bank is addressed. With devices that don't support common RAM, the W register must be provided for in each bank. The 16C745/765 can save the appropriate registers in Common RAM and not have to waste a byte in each bank for W register.

10.9.3.6 Buffer Allocation

The PIC16C745/765 has 64 bytes of Dual Port RAM. 24 are used for the Buffer Descriptor Table (BDT), leaving 40 bytes for buffers.

Endpoints 0 IN and OUT need dedicated buffers since a setup transaction can never be NAKed. That leaves three buffers for four possible Endpoints, but the USB spec requires that low speed devices are only allowed 2 endpoints (USB 1.1 paragraph 5.3.1.2), where an endpoint is a simplex connection that is defined by the combination of Endpoint number and direction.

10.9.3.7 Vendor Specific Commands

Vendor specific commands are defined by the vendor. These are parsed out, but are not processed. Instead, control is passed to function **CheckVendor** where they can be processed.

10.9.4 FILE PACKAGING

The software interface is packaged into four files, designed to simplify the integration with your application.

File **USB_CH9.ASM** contains the interface and core functions needed to enumerate the bus. **DESCRIPT.ASM** contains the device, config, interface, endpoint and string descriptors. Both of these files must be linked in with your application.

HIDCLASS.ASM provides some HID Class specific functions. Currently only **GetReportDescriptor** is supported. Other class specific functions can be implemented in a similar fashion. When a token done interrupt determines that it's a class specific command on the basis that ReportType bit 6 is set, control is passed to function **ClassSpecific**. If you're working with a different class, this is your interface between the core functions and the class specific functions.

USB_MAIN.ASM is useful as a starting point on a new application and as an example of how an existing application needs to service the USB interrupt and communicate with the core functions.

10.9.5 FUNCTION CALL REFERENCE

Interface between the Application and Protocol layer takes place in three main functions: **InitUSB, PutUSB** and **GetUSB**.

InitUSB should be called by the main program immediately upon power-up. It enables the USB peripheral and USB Reset interrupt, and transitions the part to the powered state to prepare the device for enumeration. See Section 10.9.6 “Behind the Scenes” for details on the enumeration process.

DeInitUSB disables the USB peripheral, removing the device from the bus. An application might call **DeInitUSB** if it was finished communicating to the host and didn't want to be polled any more.

PutUSB (Buffer pointer, Buffer size, Endpoint) sends data up to the host. The pointer to the block of data to transmit is in the FSR/IRP, and the block size and endpoint is passed in W register. If the IN buffer is available for that endpoint, **PutUSB** copies the buffer, flips the Data 0/1 bit and sets the OWNS bit. A buffer not available would occur when it has been previously loaded and the host has not requested that the USB peripheral transmit it. In this case, a failure code would be returned so the application can try again later.

GetUSB (Buffer Pointer, Endpoint) returns data sent from the host. If the out buffer pointed to by the endpoint number is ready, as indicated by the OWNS bit, the buffer is copied from dual port RAM to the locations pointed to by the buffer pointer, and resets the endpoint for the next out transaction from the host. If no data is available, it returns a failure code. Thus the functions of polling for buffer ready and copying the data are combined into the one function.

ServiceUSBInt handles all interrupts generated by the USB peripheral. First, it copies the active buffer to common RAM, which provides a quick turn around on the buffer in dual port RAM and also avoids having to switch banks during processing of the buffer. File **USB_MAIN.ASM** gives an example of how **ServiceUSBInt** would be invoked.

StallUSBEP/UnstallUSBEP sets or clears the stall bit in the endpoint control register. The stall bit indicates to the host that user intervention is required and until such intervention is made, further attempts to communicate with the endpoint will not be successful. Once the user intervention has been made, **UnstallUSBEP** clears the bit allowing communication to take place. These calls are useful to signal to the host that user intervention is required. An example of this might be a printer out of paper.

SoftDetachUSB clears the DEV_ATT bit, electrically disconnecting the device from the bus, then reconnecting, so it can be re-enumerated by the host. This process takes approximately 50 mS, to ensure that the host has seen the device disconnect and reattach to the bus.

CheckSleep tests the UCTRL.UIDLE bit if set, indicating that there has been no activity on the bus for 3 mS. If set, the device can be put to SLEEP, which puts the part into a low power standby mode, until wakened by bus activity. This has to be handled outside the ISR because we need the interrupt to wake us from SLEEP, and also because the application may not be ready to SLEEP when the interrupt occurs. Instead, the application should periodically call this function to poll the bit, when the device is in a good place to SLEEP.

Prior to putting the device to SLEEP, it enables the activity interrupt so the device will be awakened by the first transition on the bus. The PIC device will immediately jump to the ISR, recognize the activity interrupt, which then disables the interrupt and resumes processing with the instruction following the **CheckSleep** call.

ConfiguredUSB (Macro) continuously polls the enumeration status bits and waits until the device has been configured by the host. This should be used after the call to **InitUSB** and prior to the first time your application attempts to communicate on the bus.

SetConfiguration is a callback function that allows your application to associate some meaning to a Set Configuration command from the host. The CH9 software stores the value in USB_Curr_Config so it can be reported back on a **Get Configuration** call. This function is also called, passing the new configuration in W. This function is called from within the ISR, so it should be kept as short as possible.

10.9.6 BEHIND THE SCENES

InitUSB clears the error counters and enables the 3.3V regulator and the USB Reset interrupt. This implements the requirement to prevent the PIC device from responding to commands until the device has been RESET.

The host sees the device and resets the device, to begin the enumeration process. The RESET then initializes the Buffer Descriptor Table (BDT), EndPoint Control Registers and enables the remaining USB interrupt sources.

The Interrupt transfers control to the interrupt vector (address `0x0004`). Any Interrupt Service Routine must preserve the processor state by saving the FSRs that might change during interrupt processing. We recommend saving W, STATUS, PCLATH and FSR. W can be stored in unbanked RAM to avoid banking issues. Then it starts polling the Interrupt flags to see what triggered the interrupt. The USB interrupts are serviced by calling **ServiceUSBInt** which further tests the USB interrupt sources to determine how to process the interrupt.

Then, the host sends a setup token requesting the device descriptor. The USB Peripheral receives the Setup transaction, places the data portion in the EP0 OUT buffer, loads the USTAT register to indicate which endpoint received the data and triggers the Token Done (TOK_DNE) interrupt. The Chapter 9 commands then interpret the Setup token and sets up the data to respond to the request in the EP0 IN buffer, then sets the UOWN bit to tell the SIE there is data available.

Then, the host sends an IN transaction to receive the data from the setup transaction. The SIE sends the data from the EP0 IN buffer and then sets the Token Done interrupt to notify us that the data has been sent. If there is additional data, the next buffer is setup in EP0 IN buffer.

This token processing sequence holds true for the entire enumeration sequence, which walks through the flow chart starting chapter 9 of the USB spec. The device starts off in the powered state, transitions to default via the Reset interrupt, transitions to ADDRESSED via the **SetAddress** command, and transitions to CONFIGURED via a **SetConfiguration** command.

The USB peripheral detects several different errors and handles most internally. The USB_ERR interrupt notifies the PIC device that an error has occurred. No action is required by the device when an error occurs. Instead, the errors are simply acknowledged and counted. There is no mechanism to pull the device off the bus if there are too many errors. If this behavior is desired, it must be implemented in the application.

The Activity interrupt is left disabled until the USB peripheral detects no bus activity for 3 mS. Then it suspends the USB peripheral and enables the activity interrupt. The activity interrupt then reactivates the USB peripheral when bus activity resumes, so processing may continue.

CheckSleep is a separate call that takes the bus idle one step further and puts the PIC device to SLEEP, if the USB peripheral has detected no activity on the bus. This powers down most of the device to minimal current draw. This call should be made at a point in the main loop where all other processing is complete.

10.9.7 EXAMPLE

This example shows how the USB functions are used. This example first initializes the USB peripheral, which allows the host to enumerate the device. The enumeration process occurs in the background, via an Interrupt Service Routine. This function waits until enumeration is complete, and then polls EP1 OUT to see if there is any data available. When a buffer is available, it is copied to the IN buffer. Presumably your application would do something more interesting with the data than this example.

```
; *****************************************************************
; Demo program that initializes the USB peripheral, allows the Host
;     to Enumerate, then copies buffers from EP1OUT to EP1IN.
; *****************************************************************
main
    call     InitUSB          ; Set up everything so we can enumerate
    ConfiguredUSB             ; wait here until we have enumerated.

CheckEP1         ; Check Endpoint 1 for an OUT transaction
    bankisel buffer           ; point to lower banks
    movlw    buffer
    movwf    FSR              ; point FSR to our buffer
    movlw    1                ; check end point 1
    call     GetUSB           ; If data is ready, it will be copied.
    btfss    STATUS,C         ; was there any data for us?
    goto     PutBuffer        ; Nope, check again.
; Code host to process out buffer from host

PutBuffer
    bankisel buffer           ; point to lower banks
                              ; save buffer length
    movlw    buffer
    movwf    FSR              ; point FSR to our buffer
    movlw    0x81             ; put 8 bytes to Endpoint 1
    call     PutUSB
    btfss    STATUS,C         ; was it successful?
    goto     PutBuffer        ; No: try again until successful
    goto     idleloop         ; Yes: restart loop

    end
```

10.9.8 ASSEMBLING THE CODE

The code is designed to be used with the linker. There is no provision for includable files. The code comes packaged as several different files:

- **USB_CH9.ASM** - handles all the Chapter 9 command processing.
- **USB_DEFS.INC** - #Defines used throughout the code.
- **USB_MAIN.ASM** - Sample interrupt service routine.
- **HIDCLASS.ASM** - Handles the HID class specific commands.

10.9.8.1 Assembly Options

There are two #defines at the top of the code that control assembly options.

10.9.8.2 #define ERRORCOUNTERS

This define includes code to count the number of errors that occur, by type of error. This requires extra code and RAM locations to implement the counters.

10.9.8.3 #define FUNCTIONIDS

This is useful for debug. It encodes the upper 6 bits of USWSTAT (0x197) to indicate which function is executing. See the defines in **USB_DEFS.INC** for the codes that will be encoded.

NOTES:

11.0 UNIVERSAL SYNCHRONOUS ASYNCHRONOUS RECEIVER TRANSMITTER (USART)

The Universal Synchronous Asynchronous Receiver Transmitter (USART) module is one of the two serial I/O modules. (USART is also known as a Serial Communications Interface or SCI). The USART can be configured as a full duplex asynchronous system that can communicate with peripheral devices, such as CRT terminals and personal computers, or it can be configured as a half duplex synchronous system that can communicate with peripheral devices, such as A/D or D/A integrated circuits, Serial EEPROMs, etc.

The USART can be configured in the following modes:

- Asynchronous (full duplex)
- Synchronous - Master (half duplex)
- Synchronous - Slave (half duplex)

Bits SPEN (RCSTA<7>) and TRISC<7:6> have to be set in order to configure pins RC6/TX/CK and RC7/RX/DT as the Universal Synchronous Asynchronous Receiver transmitter.

REGISTER 11-1: TRANSMIT STATUS AND CONTROL REGISTER (TXSTA: 98h)

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D
bit7							bit0

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

bit 7: **CSRC**: Clock Source Select bit

Asynchronous mode
Don't care

Synchronous mode
1 = Master mode (Clock generated internally from BRG)
0 = Slave mode (Clock from external source)

bit 6: **TX9**: 9-bit Transmit Enable bit
1 = Selects 9-bit transmission
0 = Selects 8-bit transmission

bit 5: **TXEN**: Transmit Enable bit
1 = Transmit enabled
0 = Transmit disabled
Note: SREN/CREN overrides TXEN in SYNC mode.

bit 4: **SYNC**: USART Mode Select bit
1 = Synchronous mode
0 = Asynchronous mode

bit 3: **Unimplemented:** Read as '0'

bit 2: **BRGH**: High Baud Rate Select bit

Asynchronous mode
1 = High speed
0 = Low speed

Synchronous mode
Unused in this mode

bit 1: **TRMT**: Transmit Shift Register Status bit
1 = TSR empty
0 = TSR full

bit 0: **TX9D**: 9th bit of transmit data. (Can be used for parity.)

REGISTER 11-2: RECEIVE STATUS AND CONTROL REGISTER (RCSTA: 18h)

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D
bit7							bit0

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

bit 7: **SPEN**: Serial Port Enable bit
1 = Serial port enabled (Configures RC7/RX/DT and RC6/TX/CK pins as serial port pins)
0 = Serial port disabled

bit 6: **RX9**: 9-bit Receive Enable bit
1 = Selects 9-bit reception
0 = Selects 8-bit reception

bit 5: **SREN**: Single Receive Enable bit

Asynchronous mode
Don't care

Synchronous mode - master
1 = Enables single receive
0 = Disables single receive
This bit is cleared after reception is complete.

Synchronous mode - slave
Unused in this mode

bit 4: **CREN**: Continuous Receive Enable bit

Asynchronous mode
1 = Enables continuous receive
0 = Disables continuous receive

Synchronous mode
1 = Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN)
0 = Disables continuous receive

bit 3: **Unimplemented:** Read as '0'

bit 2: **FERR**: Framing Error bit
1 = Framing error (Can be updated by reading RCREG register and receive next valid byte)
0 = No framing error

bit 1: **OERR**: Overrun Error bit
1 = Overrun error (Can be cleared by clearing bit CREN)
0 = No overrun error

bit 0: **RX9D**: 9th bit of received data. (Can be used for parity.)

11.1 USART Baud Rate Generator (BRG)

The BRG supports both the Asynchronous and Synchronous modes of the USART. It is a dedicated 8-bit baud rate generator. The SPBRG register controls the period of a free running 8-bit timer. In Asynchronous mode, bit BRGH (TXSTA<2>) also controls the baud rate. In Synchronous mode, bit BRGH is ignored. Table 11-1 shows the formula for computation of the baud rate for different USART modes which only apply in Master mode (internal clock).

Given the desired baud rate and FINT, the nearest integer value for the SPBRG register can be calculated using the formula in Table 11-1. From this, the error in baud rate can be determined.

It may be advantageous to use the high baud rate (BRGH = 1) even for slower baud clocks. This is because the FINT/(16(X + 1)) equation can reduce the baud rate error in some cases.

Writing a new value to the SPBRG register causes the BRG timer to be reset (or cleared). This ensures the BRG does not wait for a timer overflow before outputting the new baud rate.

11.1.1 SAMPLING

The data on the RC7/RX/DT pin is sampled three times near the center of each bit time by a majority detect circuit to determine if a high or a low level is present at the RX pin.

TABLE 11-1: BAUD RATE FORMULA

SYNC	BRGH = 0 (Low Speed)	BRGH = 1 (High Speed)
0	(Asynchronous) Baud Rate = FINT/(64(SPBRG+1))	Baud Rate = FINT/(16(SPBRG+1))
1	(Synchronous) Baud Rate = FINT/(4(SPBRG+1))	NA

TABLE 11-2: BAUD RATES FOR SYNCHRONOUS MODE

Desired Baud	24 MHz		
	Actual Baud	% of Error	SPBRG
300	—	—	—
1200	—	—	—
2400	—	—	—
4800	—	—	—
9600	—	—	—
19200	—	—	—
38400	38461.54	0.16	155
57600	57692.31	0.16	103
115200	115384.62	0.16	51
230400	230769.23	0.16	25
460800	461538.46	0.16	12
921600	1000000.00	8.51	5

TABLE 11-3: BAUD RATES FOR ASYNCHRONOUS MODE (BRGH = 0)

Desired Baud	24 MHz		
	Actual Baud	% of Error	SPBRG
300	—	—	—
1200	—	—	—
2400	2403.85	0.16	155
4800	4807.69	0.16	77
9600	9615.38	0.16	38
19200	19736.84	2.80	18
38400	41666.67	8.51	8
57600	62500.00	8.51	5
115200	125000.00	8.51	2
230400	—	—	—
460800	—	—	—
921600	—	—	—

TABLE 11-4: BAUD RATES FOR ASYNCHRONOUS MODE (BRGH = 1)

Desired Baud	24 MHz		
	Actual Baud	% of Error	SPBRG
300	—	—	—
1200	—	—	—
2400	—	—	—
4800	—	—	—
9600	9615.38	0.16	155
19200	19230.77	0.16	77
38400	38461.54	0.16	38
57600	57692.31	0.16	25
115200	115384.62	0.16	12
230400	250000.00	8.51	5
460800	500000.00	8.51	2
921600	—	—	—

TABLE 11-5: REGISTERS ASSOCIATED WITH BAUD RATE GENERATOR

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
18h	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

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

11.2 USART Asynchronous Mode

In this mode, the USART uses standard nonreturn-to-zero (NRZ) format (one start bit, eight or nine data bits, and one stop bit). The most common data format is 8 bits. An on-chip, dedicated, 8-bit baud rate generator can be used to derive standard baud rate frequencies from the oscillator. The USART transmits and receives the LSb first. The USART's transmitter and receiver are functionally independent, but use the same data format and baud rate. The baud rate generator produces a clock either x16 or x64 of the bit shift rate, depending on bit BRGH (TXSTA<2>). Parity is not supported by the hardware, but can be implemented in software (and stored as the ninth data bit). Asynchronous mode is stopped during SLEEP.

Asynchronous mode is selected by clearing bit SYNC (TXSTA<4>).

The USART Asynchronous module consists of the following important elements:

- Baud Rate Generator
- Sampling Circuit
- Asynchronous Transmitter
- Asynchronous Receiver

11.2.1 USART ASYNCHRONOUS TRANSMITTER

The USART transmitter block diagram is shown in Figure 11-1. The heart of the transmitter is the transmit (serial) shift register (TSR). The shift register obtains its data from the read/write transmit buffer, TXREG. The TXREG register is loaded with data in software. The TSR register is not loaded until the STOP bit has been transmitted from the previous load. As soon as the STOP bit is transmitted, the TSR is loaded with new data from the TXREG register (if available). Once the TXREG register transfers the data to the TSR register (occurs in one TCY), the TXREG register is empty and flag bit TXIF (PIR1<4>) is set. This interrupt can be enabled/disabled by setting/clearing enable bit TXIE (PIE1<4>). Flag bit TXIF will be set regardless of the state of enable bit TXIE and cannot be cleared in software. It will reset only when new data is loaded into the TXREG register. While flag bit TXIF indicated the status of the TXREG register, another bit TRMT (TXSTA<1>) shows the status of the TSR register. Status bit TRMT is a read only bit, which is set when the TSR register is empty. No interrupt logic is tied to this bit, so the user has to poll this bit in order to determine if the TSR register is empty.

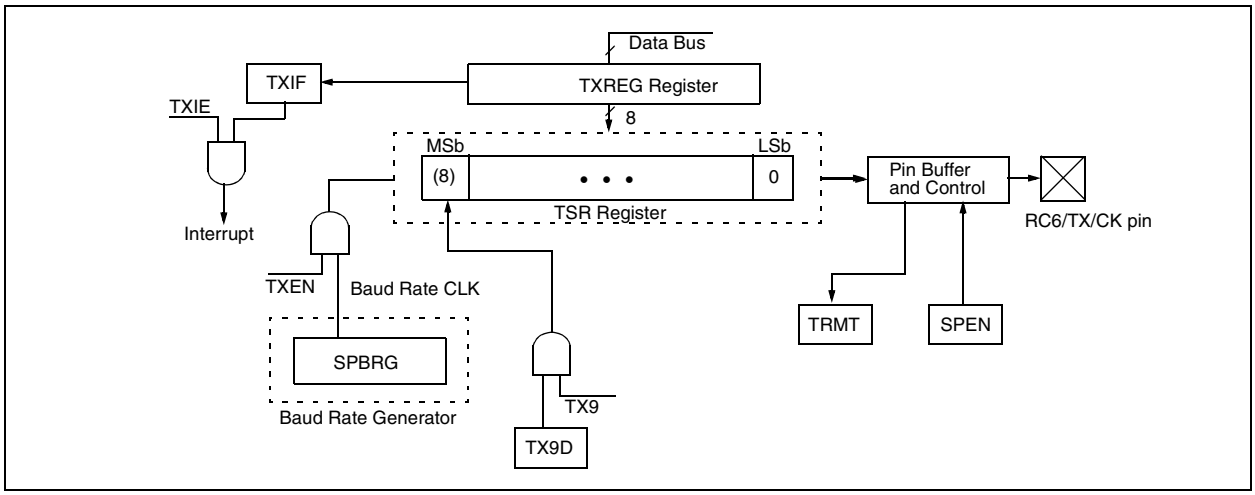
Note 1: The TSR register is not mapped in data memory, so it is not available to the user.

2: Flag bit TXIF is set when enable bit TXEN is set. TXIF is cleared by loading TXREG.

Transmission is enabled by setting enable bit TXEN (TXSTA<5>). The actual transmission will not occur until the TXREG register has been loaded with data and the baud rate generator (BRG) has produced a shift clock (Figure 11-2). The transmission can also be started by first loading the TXREG register and then setting enable bit TXEN. Normally, when transmission is first started, the TSR register is empty. At that point, transfer to the TXREG register will result in an immediate transfer to TSR, resulting in an empty TXREG. A back-to-back transfer is thus possible (Figure 11-3). Clearing enable bit TXEN during a transmission will cause the transmission to be aborted and will reset the transmitter. As a result, the RC6/TX/CK pin will revert to hi-impedance.

In order to select 9-bit transmission, transmit bit TX9 (TXSTA<6>) should be set and the ninth bit should be written to TX9D (TXSTA<0>). The ninth bit must be written before writing the 8-bit data to the TXREG register. This is because a data write to the TXREG register can result in an immediate transfer of the data to the TSR register (if the TSR is empty). In such a case, an incorrect ninth data bit may be loaded in the TSR register.

FIGURE 11-1: USART TRANSMIT BLOCK DIAGRAM



Steps to follow when setting up an Asynchronous Transmission:

1. Initialize the SPBRG register for the appropriate baud rate. If a high speed baud rate is desired, set bit BRGH. (Section 11.1)
2. Enable the asynchronous serial port by clearing bit SYNC and setting bit SPEN.
3. If interrupts are desired, then set enable bit TXIE.
4. If 9-bit transmission is desired, then set transmit bit TX9.
5. Enable the transmission by setting bit TXEN, which will also set bit TXIF.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit TX9D.
7. Load data to the TXREG register (starts transmission).

FIGURE 11-2: ASYNCHRONOUS MASTER TRANSMISSION

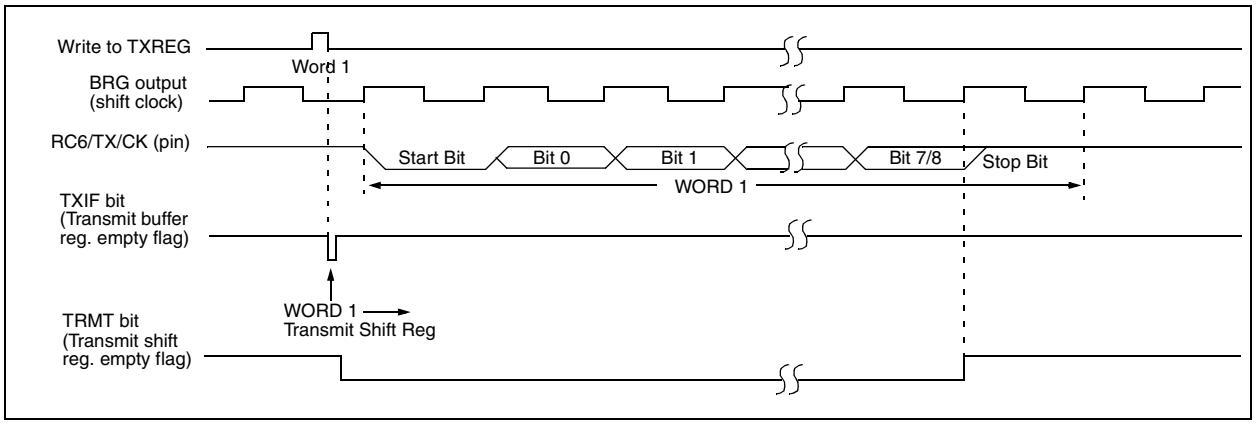
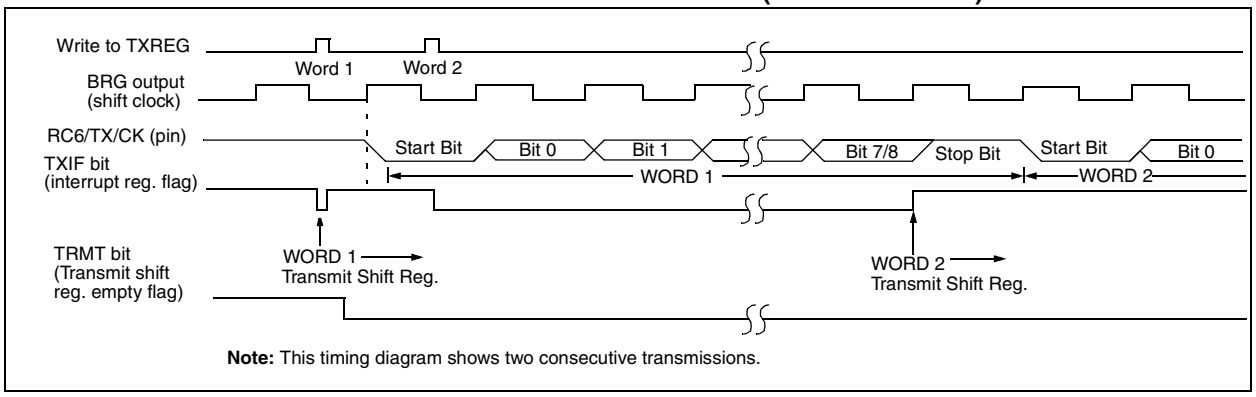


FIGURE 11-3: ASYNCHRONOUS MASTER TRANSMISSION (BACK TO BACK)



Note: This timing diagram shows two consecutive transmissions.

TABLE 11-6: REGISTERS ASSOCIATED WITH ASYNCHRONOUS TRANSMISSION

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	USBIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
19h	TXREG	USART Transmit Register								0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	USBIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented locations read as '0'. Shaded cells are not used for asynchronous transmission.
Note 1: Bits PSPIE and PSPIF are reserved on the PIC16C745; always maintain these bits clear.

11.2.2 USART ASYNCHRONOUS RECEIVER

The receiver block diagram is shown in Figure 11-4. The data is received on the RC7/RX/DT pin and drives the data recovery block. The data recovery block is actually a high speed shifter operating at x16 times the baud rate, whereas the main receive serial shifter operates at the bit rate or at FINT.

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

The heart of the receiver is the receive (serial) shift register (RSR). After sampling the STOP bit, the received data in the RSR is transferred to the RCREG register (if it is empty). If the transfer is complete, flag bit RCIF (PIR1<5>) is set. The actual interrupt can be enabled/disabled by setting/clearing enable bit RCIE (PIE1<5>). Flag bit RCIF is a read only bit which is cleared by the hardware. It is cleared when the RCREG register has been read and is empty. The RCREG is a double buffered register, i.e., it is a two deep FIFO. It is possible for two bytes of data to be received and transferred to the RCREG FIFO and a third byte to begin shifting to the RSR register. On the detection of the STOP bit of the third byte, if the RCREG register is still full, then overrun error bit OERR (RCSTA<1>) will be set. The word in the RSR will be lost. The RCREG register can be read twice to retrieve the two bytes in the FIFO. Overrun bit OERR has to be cleared in software. This is done by resetting the receive logic (CREN is cleared and then set). If bit OERR is set, transfers from the RSR register to the RCREG register are inhibited, so it is essential to clear error bit OERR if it is set. Framing error bit FERR (RCSTA<2>) is set if a stop bit is detected as clear. Bit FERR and the 9th receive bit are buffered the same way as the receive data. Reading the RCREG, will load bits RX9D and FERR with new values, therefore it is essential for the user to read the RCSTA register before reading RCREG register in order not to lose the old FERR and RX9D information.

FIGURE 11-4: USART RECEIVE BLOCK DIAGRAM

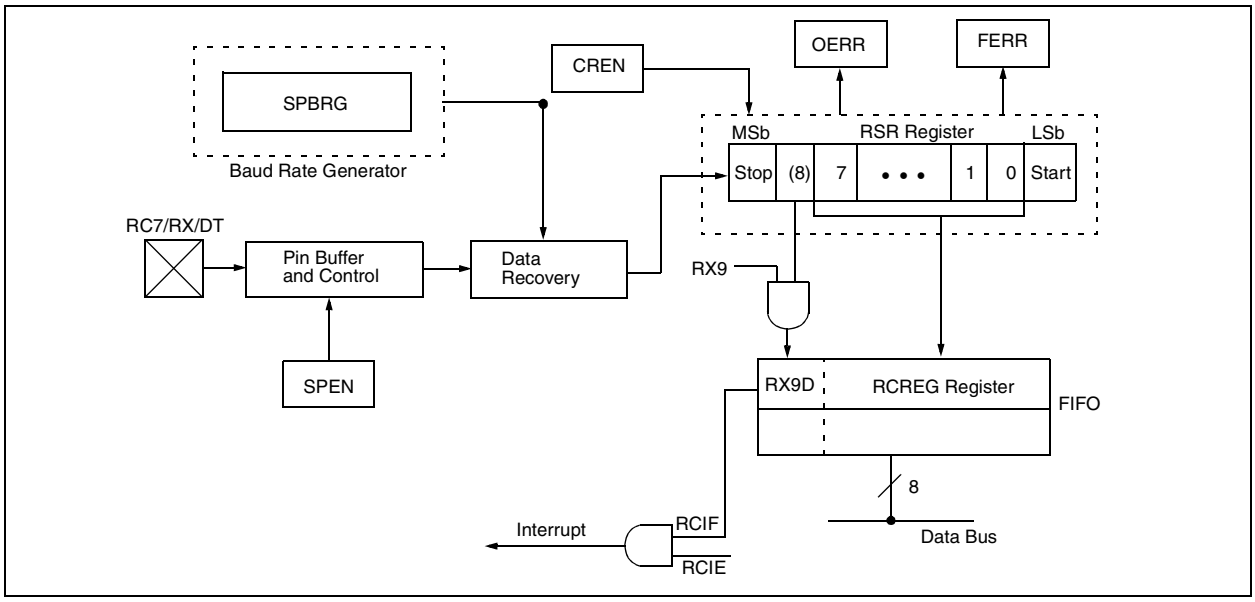
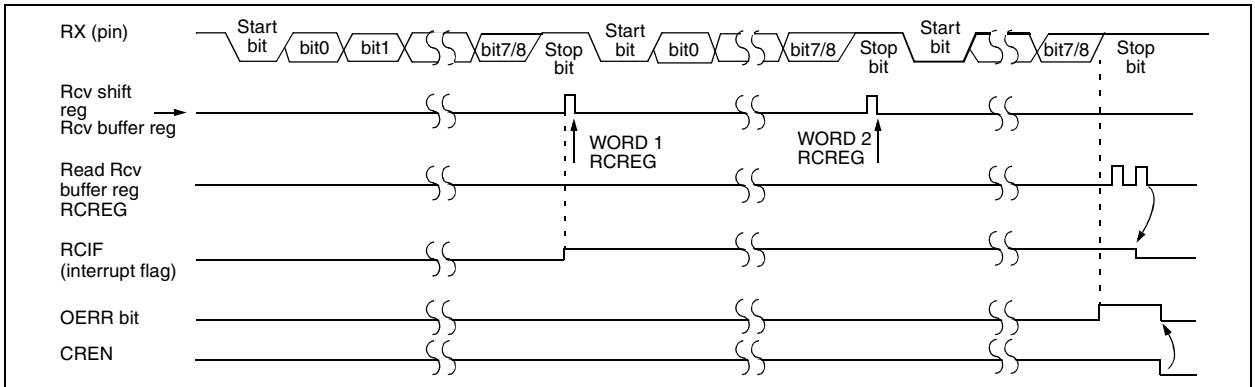


FIGURE 11-5: ASYNCHRONOUS RECEPTION



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

Steps to follow when setting up an Asynchronous Reception:

1. Initialize the SPBRG register for the appropriate baud rate. If a high speed baud rate is desired, set bit BRGH. (Section 11.1).
2. Enable the asynchronous serial port by clearing bit SYNC, and setting bit SPEN.
3. If interrupts are desired, then set enable bit RCIE.
4. If 9-bit reception is desired, then set bit RX9.
5. Enable the reception by setting bit CREN.
6. Flag bit RCIF will be set when reception is complete and an interrupt will be generated if enable bit RCIE was set.
7. Read the RCSTA register to get the ninth bit (if enabled) and determine if any error occurred during reception.
8. Read the 8-bit received data by reading the RCREG register.
9. If any error occurred, clear the error by clearing enable bit CREN.

TABLE 11-7: REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	USBIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
1Ah	RCREG	USART Receive Register								0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	USBIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented locations read as '0'. Shaded cells are not used for asynchronous reception.
Note 1: Bits PSPIE and PSPIF are reserved on the PIC16C745; always maintain these bits clear.

11.3 USART Synchronous Master Mode

In Synchronous Master mode, the data is transmitted in a half-duplex manner, i.e., transmission and reception do not occur at the same time. When transmitting data, the reception is inhibited and vice versa. Synchronous mode is entered by setting bit SYNC (TXSTA<4>). In addition, enable bit SPEN (RCSTA<7>) is set in order to configure the RC6/TX/CK and RC7/RX/DT I/O pins to CK (clock) and DT (data) lines, respectively. The Master mode indicates that the processor transmits the master clock on the CK line. The Master mode is entered by setting bit CSRC (TXSTA<7>).

11.3.1 USART SYNCHRONOUS MASTER TRANSMISSION

The USART transmitter block diagram is shown in Figure 11-1. The heart of the transmitter is the transmit (serial) shift register (TSR). The shift register obtains its data from the read/write transmit buffer register TXREG. The TXREG register is loaded with data in software. The TSR register is not loaded until the last bit has been transmitted from the previous load. As soon as the last bit is transmitted, the TSR is loaded with new data from the TXREG (if available). Once the TXREG register transfers the data to the TSR register (occurs in one Tcycle), the TXREG is empty and interrupt bit TXIF (PIR1<4>) is set. The interrupt can be enabled/disabled by setting/clearing enable bit TXIE (PIE1<4>). Flag bit TXIF will be set regardless of the state of enable bit TXIE and cannot be cleared in software. It will reset only when new data is loaded into the TXREG register. While flag bit TXIF indicates the status of the TXREG register, another bit TRMT (TXSTA<1>) shows the status of the TSR register. TRMT is a read only bit which is set when the TSR is empty. No interrupt logic is tied to this bit, so the user has to poll this bit in order to determine if the TSR register is empty. The TSR is not mapped in data memory, so it is not available to the user.

Transmission is enabled by setting enable bit TXEN (TXSTA<5>). The actual transmission will not occur until the TXREG register has been loaded with data. The first data bit will be shifted out on the next available rising edge of the clock on the CK line. Data out is stable around the falling edge of the synchronous clock (Figure 11-6). The transmission can also be started by first loading the TXREG register and then setting bit TXEN (Figure 11-7). This is advantageous when slow baud rates are selected, since the BRG is kept in RESET when bits TXEN, CREN and SREN are clear. Setting enable bit TXEN will start the BRG, creating a shift clock immediately. Normally, when transmission is first started, the TSR register is empty, so a transfer to the TXREG register will result in an immediate transfer to TSR resulting in an empty TXREG. Back-to-back transfers are possible.

Clearing enable bit TXEN, during a transmission, will cause the transmission to be aborted and will reset the transmitter. The DT and CK pins will revert to hi-impedance. If either bit CREN or bit SREN is set during a transmission, the transmission is aborted and the DT pin reverts to a hi-impedance state (for a reception). The CK pin will remain an output if bit CSRC is set (internal clock). The transmitter logic, however, is not reset, although it is disconnected from the pins. In order to reset the transmitter, the user has to clear bit TXEN. If bit SREN is set (to interrupt an on-going transmission and receive a single word), then after the single word is received, bit SREN will be cleared and the serial port will revert back to transmitting, since bit TXEN is still set. The DT line will immediately switch from hi-impedance receive mode to transmit and start driving. To avoid this, bit TXEN should be cleared.

In order to select 9-bit transmission, the TX9 (TXSTA<6>) bit should be set and the ninth bit should be written to bit TX9D (TXSTA<0>). The ninth bit must be written before writing the 8-bit data to the TXREG register. This is because a data write to the TXREG can result in an immediate transfer of the data to the TSR register (if the TSR is empty). If the TSR was empty and the TXREG was written before writing the “new” TX9D, the “present” value of bit TX9D is loaded.

Steps to follow when setting up a Synchronous Master Transmission:

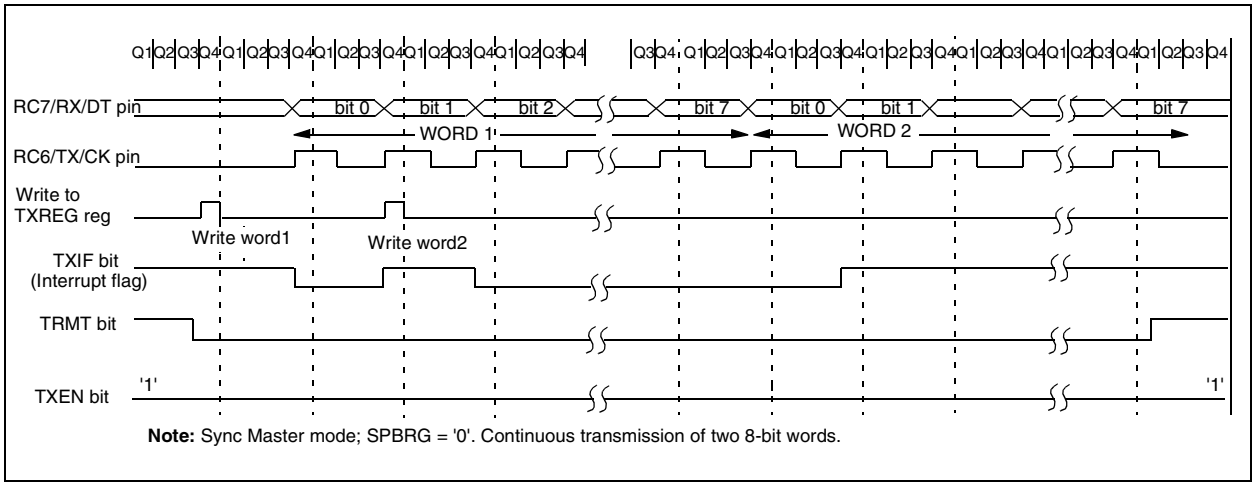
1. Initialize the SPBRG register for the appropriate baud rate (Section 11.1).
2. Enable the synchronous master serial port by setting bits SYNC, SPEN and CSRC.
3. If interrupts are desired, set enable bit TXIE.
4. If 9-bit transmission is desired, set bit TX9.
5. Enable the transmission by setting bit TXEN.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit TX9D.
7. Start transmission by loading data to the TXREG register.

TABLE 11-8: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER TRANSMISSION

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	USBIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
19h	TXREG	USART Transmit Register								0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	USBIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

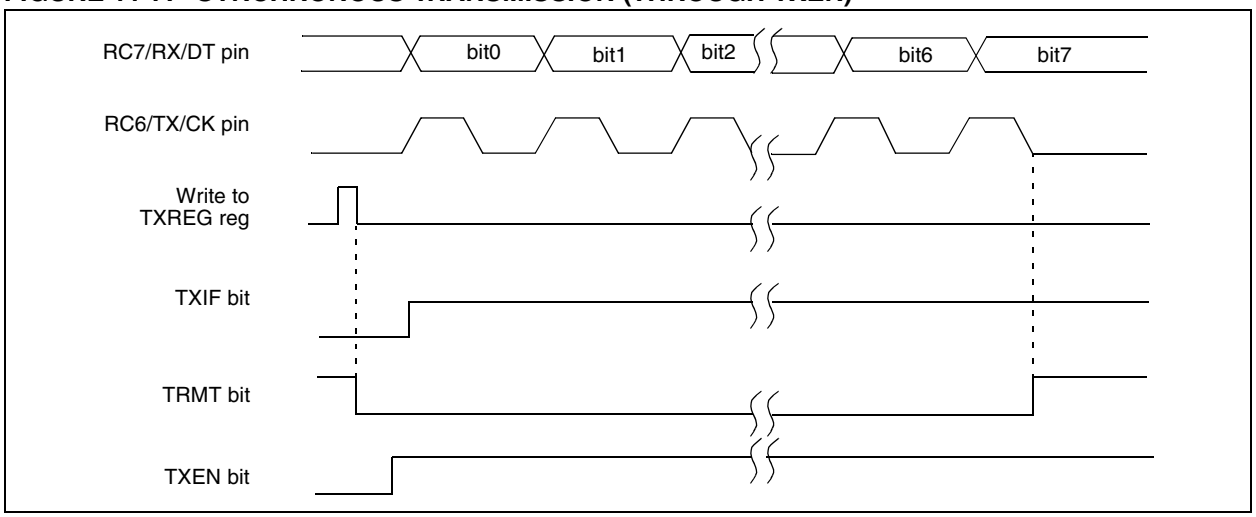
Legend: x = unknown, - = unimplemented, read as '0'. Shaded cells are not used for synchronous master transmission.
Note 1: Bits PSPIE and PSPIF are reserved on the PIC16C745; always maintain these bits clear.

FIGURE 11-6: SYNCHRONOUS TRANSMISSION



Note: Sync Master mode; SPBRG = '0'. Continuous transmission of two 8-bit words.

FIGURE 11-7: SYNCHRONOUS TRANSMISSION (THROUGH TXEN)



11.3.2 USART SYNCHRONOUS MASTER RECEPTION

Once Synchronous mode is selected, reception is enabled by setting either enable bit SREN (RCSTA<5>) or enable bit CREN (RCSTA<4>). Data is sampled on the RC7/RX/DT pin on the falling edge of the clock. If enable bit SREN is set, then only a single word is received. If enable bit CREN is set, the reception is continuous until CREN is cleared. If both bits are set, CREN takes precedence. After clocking the last bit, the received data in the Receive Shift Register (RSR) is transferred to the RCREG register (if it is empty). When the transfer is complete, interrupt flag bit RCIF (PIR1<5>) is set. The actual interrupt can be enabled/disabled by setting/clearing enable bit RCIE (PIE1<5>). Flag bit RCIF is a read only bit, which is reset by the hardware. In this case, it is reset when the RCREG register has been read and is empty. The RCREG is a double buffered register, i.e., it is a two deep FIFO. It is possible for two bytes of data to be received and transferred to the RCREG FIFO and a third byte to begin shifting into the RSR register. On the clocking of the last bit of the third byte, if the RCREG register is still full, then overrun error bit OERR (RCSTA<1>) is set. The word in the RSR will be lost. The RCREG register can be read twice to retrieve the two bytes in the FIFO. Bit OERR has to be cleared in software (by clearing bit CREN). If bit OERR is set, transfers from the RSR to the RCREG are inhibited, so it is essential to clear bit OERR if it is set. The ninth receive bit is buffered the same way as the receive data. Reading the RCREG register will load bit RX9D with a new value, therefore it is essential for the user to read the RCSTA register before reading RCREG in order not to lose the old RX9D information.

Steps to follow when setting up a Synchronous Master Reception:

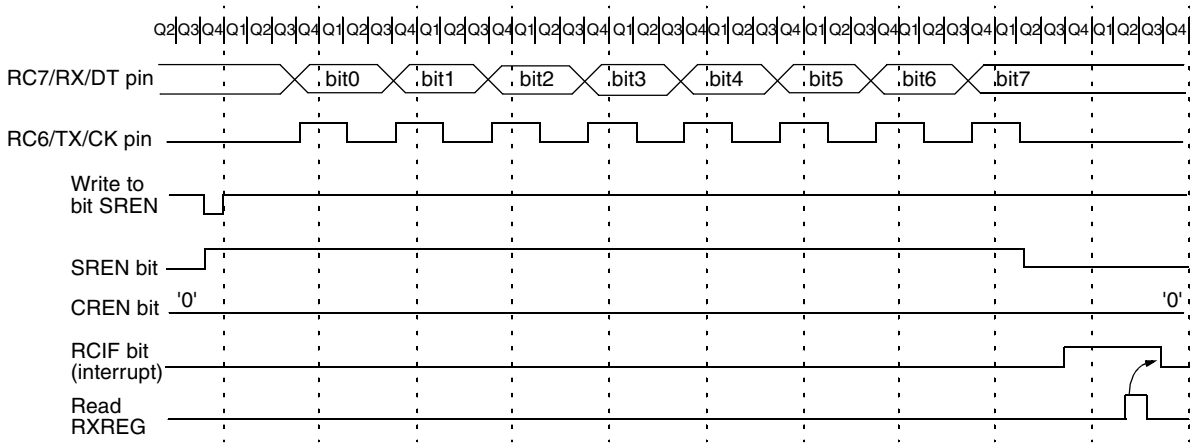
1. Initialize the SPBRG register for the appropriate baud rate (Section 11.1).
2. Enable the synchronous master serial port by setting bits SYNC, SPEN, and CSRC.
3. Ensure bits CREN and SREN are clear.
4. If interrupts are desired, then set enable bit RCIE.
5. If 9-bit reception is desired, then set bit RX9.
6. If a single reception is required, set bit SREN. For continuous reception set bit CREN.
7. Interrupt flag bit RCIF will be set when reception is complete and an interrupt will be generated if enable bit RCIE was set.
8. Read the RCSTA register to get the ninth bit (if enabled) and determine if any error occurred during reception.
9. Read the 8-bit received data by reading the RCREG register.
10. If any error occurred, clear the error by clearing bit CREN.

TABLE 11-9: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER RECEPTION

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	USBIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
1Ah	RCREG	USART Receive Register								0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	USBIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented read as '0'. Shaded cells are not used for synchronous master reception.
Note 1: Bits PSPIE and PSPIF are reserved on the PIC16C745; always maintain these bits clear.

FIGURE 11-8: SYNCHRONOUS RECEPTION (MASTER MODE, SREN)



Note: Timing diagram demonstrates Sync Master mode with bit SREN = '1' and bit BRG = '0'.

11.4 USART Synchronous Slave Mode

Synchronous Slave mode differs from the Master mode in the fact that the shift clock is supplied externally at the RC6/TX/CK pin (instead of being supplied internally in Master mode). This allows the device to transfer or receive data while in SLEEP mode. Slave mode is entered by clearing bit CSRC (TXSTA<7>).

11.4.1 USART SYNCHRONOUS SLAVE TRANSMIT

The operation of the Synchronous Master and Slave modes are identical, except in the case of the SLEEP mode.

If two words are written to the TXREG and then the SLEEP instruction is executed, the following will occur:

a) The first word will immediately transfer to the TSR register and transmit.
b) The second word will remain in TXREG register.
c) Flag bit TXIF will not be set.
d) When the first word has been shifted out of TSR, the TXREG register will transfer the second word to the TSR and flag bit TXIF will now be set.
e) If enable bit TXIE is set, the interrupt will wake the chip from SLEEP and if the global interrupt is enabled, the program will branch to the interrupt vector (0004h).

Steps to follow when setting up a Synchronous Slave Transmission:

1. Enable the synchronous slave serial port by setting bits SYNC and SPEN and clearing bit CSRC.
2. Clear bits CREN and SREN.
3. If interrupts are desired, then set enable bit TXIE.
4. If 9-bit transmission is desired, set bit TX9.
5. Enable the transmission by setting enable bit TXEN.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit TX9D.
7. Start transmission by loading data to the TXREG register.

11.4.2 USART SYNCHRONOUS SLAVE RECEPTION

The operation of the Synchronous Master and Slave modes is identical, except in the case of the SLEEP mode. Also, bit SREN is a don't care in Slave mode.

If receive is enabled by setting bit CREN prior to the SLEEP instruction, a word may be received during SLEEP. On completely receiving the word, the RSR register will transfer the data to the RCREG register and if enable bit RCIE bit is set, the interrupt generated will wake the chip from SLEEP. If the global interrupt is enabled, the program will branch to the interrupt vector (0004h).

Steps to follow when setting up a Synchronous Slave Reception:

1. Enable the synchronous master serial port by setting bits SYNC and SPEN and clearing bit CSRC.
2. If interrupts are desired, set enable bit RCIE.
3. If 9-bit reception is desired, set bit RX9.
4. To enable reception, set enable bit CREN.
5. Flag bit RCIF will be set when reception is complete and an interrupt will be generated, if enable bit RCIE was set.
6. Read the RCSTA register to get the ninth bit (if enabled) and determine if any error occurred during reception.
7. Read the 8-bit received data by reading the RCREG register.
8. If any error occurred, clear the error by clearing bit CREN.

TABLE 11-10: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE TRANSMISSION

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	USBIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
19h	TXREG	USART Transmit Register								0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	USBIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented read as '0'. Shaded cells are not used for synchronous slave transmission.
Note 1: Bits PSPIE and PSPIF are reserved on the PIC16C745; always maintain these bits clear.

TABLE 11-11: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE RECEPTION

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	USBIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
1Ah	RCREG	USART Receive Register								0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	USBIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented read as '0'. Shaded cells are not used for synchronous slave reception.
Note 1: Bits PSPIE and PSPIF are reserved on the PIC16C745; always maintain these bits clear.

12.0 ANALOG-TO-DIGITAL CONVERTER (A/D) MODULE

The 8-bit Analog-To-Digital (A/D) converter module has five inputs for the PIC16C745 and eight for the PIC16C765.

The A/D allows conversion of an analog input signal to a corresponding 8-bit digital value. The output of the sample and hold is the input into the converter, which generates the result via successive approximation. The analog reference voltage is software selectable to either the device's positive supply voltage (V_{DD}) or the voltage level on the RA3/AN3/V_{REF} pin.

The A/D converter has a unique feature of being able to operate while the device is in SLEEP mode. To operate in sleep, the A/D conversion clock must be derived from the A/D's dedicated internal RC oscillator.

The A/D module has three registers. These registers are:

- A/D Result Register (ADRES)
- A/D Control Register 0 (ADCON0)
- A/D Control Register 1 (ADCON1)

The ADCON0 register, shown in Register 12-1, controls the operation of the A/D module. The ADCON1 register, shown in Register 12-2, configures the functions of the port pins. The port pins can be configured as analog inputs (RA3 can also be a voltage reference) or as digital I/O.

Additional information on using the A/D module can be found in the PIC Mid-Range MCU Family Reference Manual (DS33023) and in Application Note, AN546.

Note:	In order to maintain 8-bit A/D accuracy, ADCS<1:0> must be set to either F_{INT}/32 or F_{RC}. Choosing F_{INT}/8 or F_{INT}/2 will cause loss of accuracy, due to the USB module's requirement of running at 24 MHz.

REGISTER 12-1: A/D CONTROL REGISTER (ADCON0: 1Fh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	—	ADON
bit7							bit0

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n = Value at POR Reset

bit 7-6: **ADCS<1:0>:** A/D Conversion Clock Select bits
00 = FINT/2
01 = FINT/8
10 = FINT/32**(2)**
11 = FRC (clock derived from dedicated internal oscillator)**(2)**

bit 5-3: **CHS<2:0>**: Analog Channel Select bits
000 = channel 0, (RA0/AN0)
001 = channel 1, (RA1/AN1)
010 = channel 2, (RA2/AN2)
011 = channel 3, (RA3/AN3)
100 = channel 4, (RA5/AN4)
101 = channel 5, (RE0/AN5)**(1)**
110 = channel 6, (RE1/AN6)**(1)**
111 = channel 7, (RE2/AN7)**(1)**

bit 2: **GO/$\overline{\text{DONE}}$:** A/D Conversion Status bit

If ADON = 1
1 = A/D conversion in progress (setting this bit starts the A/D conversion)
0 = A/D conversion not in progress (This bit is automatically cleared by hardware when the A/D conversion is complete)

bit 1: **Unimplemented**: Read as '0'

bit 0: **ADON**: A/D On bit
1 = A/D converter module is operating
0 = A/D converter module is shutoff and consumes no operating current

Note 1: A/D channels 5, 6 and 7 are implemented on the PIC16C765 only.
2: Choose FINT/32 or FRC to maintain 8-bit A/D accuracy at 24 MHz.

REGISTER 12-2: A/D CONTROL REGISTER 1 (ADCON1: 9Fh)

U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
—	—	—	—	—	PCFG2	PCFG1	PCFG0
bit7							bit0

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n = Value at POR Reset

bit 7-3: **Unimplemented:** Read as '0'

bit 2-0: **PCFG<2:0>:** A/D Port Configuration Control bits

PCFG<2:0>	AN7(1)	AN6	AN5	AN4	AN3	AN2	AN1	AN0	VREF
000	A	A	A	A	A	A	A	A	VDD
001	A	A	A	A	VREF	A	A	A	RA3
010	D	D	D	A	A	A	A	A	VDD
011	D	D	D	A	VREF	A	A	A	AN3
100	D	D	D	D	A	D	A	A	VDD
101	A	A	A	A	VREF	A	A	A	AN3
11x	D	D	D	D	D	D	D	D	VDD

A = Analog input

D = Digital I/O

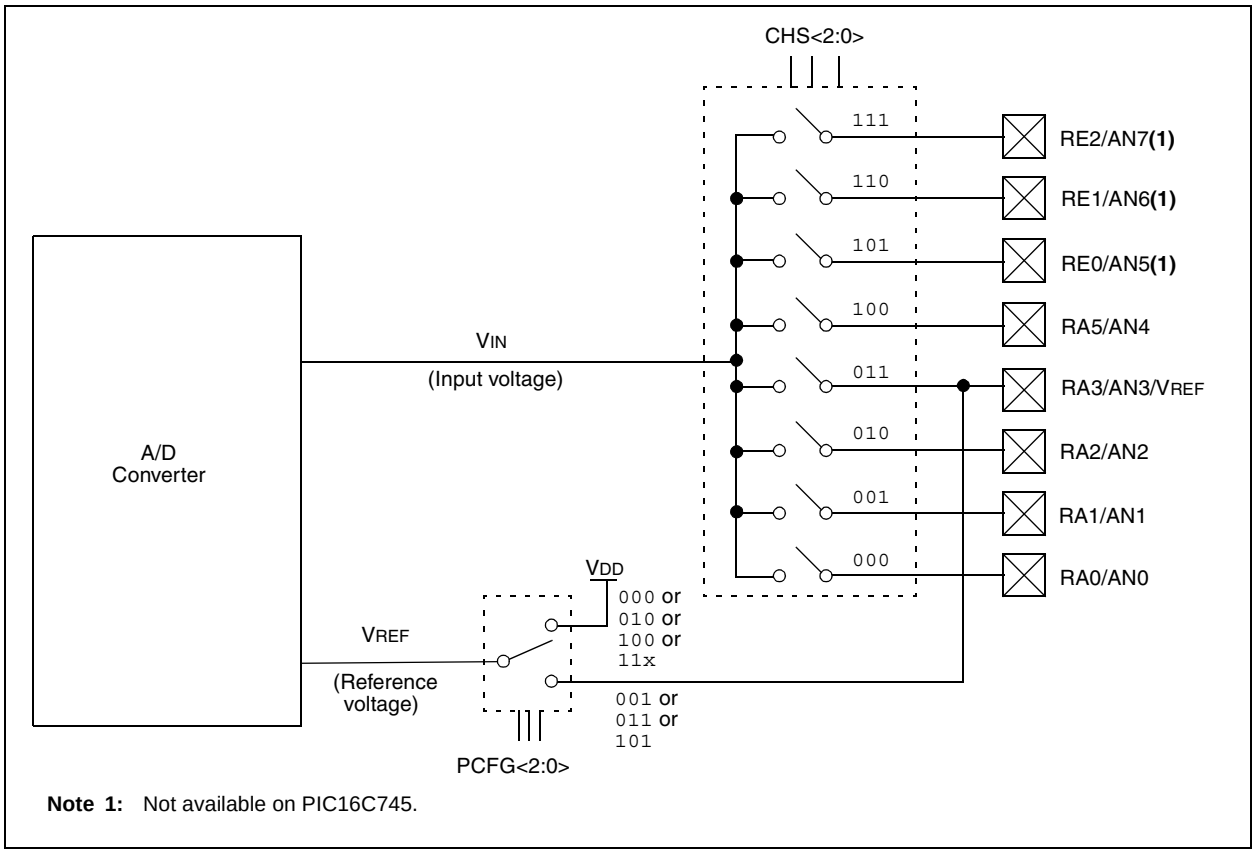
Note 1: A/D channels 5, 6 and 7 are implemented on the PIC16C765 only.

The following steps should be followed for doing an A/D conversion:

1. Configure the A/D module:
   - Configure analog pins / voltage reference / and digital I/O (ADCON1)
   - Select A/D input channel (ADCON0)
   - Select A/D conversion clock (ADCON0)
   - Turn on A/D module (ADCON0)
2. Configure A/D interrupt (if desired):
   - Clear ADIF bit
   - Set ADIE bit
   - Set GIE bit
3. Wait the required acquisition time.
4. Start conversion:
   - Set GO/$\overline{\text{DONE}}$ bit (ADCON0)
5. Wait for A/D conversion to complete, by either:
   - Polling for the GO/$\overline{\text{DONE}}$ bit to be cleared

   OR
   - Waiting for the A/D interrupt
6. Read A/D result register (ADRES), clear bit ADIF if required.
7. For next conversion, go to step 1 or step 2 as required. The A/D conversion time per bit is defined as TAD. A minimum wait of 2TAD is required before next acquisition starts.

FIGURE 12-1: A/D BLOCK DIAGRAM



Note 1: Not available on PIC16C745.

12.1 A/D Acquisition Requirements

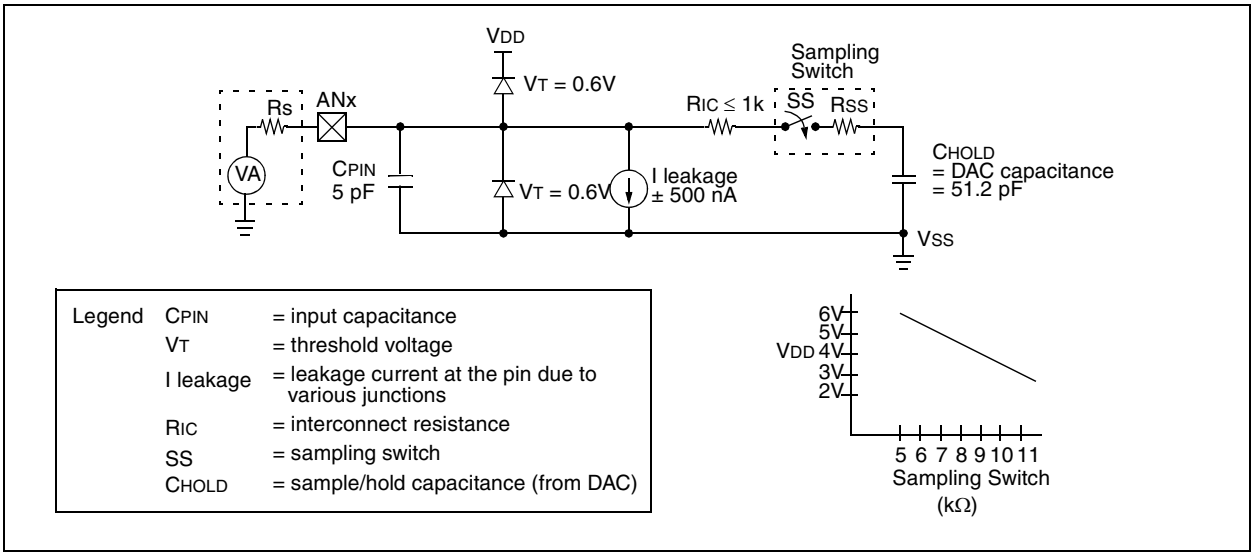
For the A/D converter to meet its specified accuracy, the charge holding capacitor (CHOLD) must be allowed to fully charge to the input channel voltage level. The analog input model is shown in Figure 12-2. The source impedance (RS) and the internal sampling switch (RSS) impedance directly affect the time required to charge the capacitor CHOLD. The sampling switch (RSS) impedance varies over the device voltage (VDD), Figure 12-2. The source impedance affects the offset voltage at the analog input (due to pin leakage current).

The maximum recommended impedance for analog sources is 10 kΩ. After the analog input channel is selected (changed), the acquisition must pass before the conversion can be started.

To calculate the minimum acquisition time, Equation 12-1 may be used. This equation assumes that 1/2 LSb error is used (512 steps for the A/D). The 1/2 LSb error is the maximum error allowed for the A/D to meet its specified resolution.

To calculate the minimum acquisition time, TACQ, see the PIC Mid-Range MCU Family Reference Manual (DS33023). In general, however, given a max of 10kΩ and a worst case temperature of 100°C, TACQ will be no more than 16 sec.

FIGURE 12-2: ANALOG INPUT MODEL



EQUATION 12-1: ACQUISITION TIME

TACQ = Amplifier Settling Time +
Hold Capacitor Charging Time +
Temperature Coefficient

= TAMP + TC + TCOFF
TAMP = 5μS
TC = - (51.2pF)(1kΩ + RSS + RS) In(1/511)
TCOFF = (Temp -25°C)(0.05μS/°C)

12.2 Selecting the A/D Conversion Clock

The A/D conversion time per bit is defined as TAD. The A/D conversion requires 9.5TAD per 8-bit conversion. The source of the A/D conversion clock is software selectable. The four possible options for TAD are:

- 2TOSC
- 8TOSC
- 32TOSC
- Dedicated Internal RC oscillator

For correct A/D conversions, the A/D conversion clock (TAD) must be selected to ensure a minimum TAD time of 1.6 μs.

TABLE 12-1: TAD vs. DEVICE OPERATING FREQUENCIES

AD Clock Source (TAD)		Device Frequency
Operation	**ADCS1:ADCS0**	**24 MHz**
2TOSC	00	83.3 ns
8TOSC	01	333.3 ns
32TOSC	10	1.333 μ
RC	11	2 - 6 μs(1,2)

Note 1: The RC source has a typical TAD time of 4 μs.
2: For device frequencies above 1 MHz, the device must be in SLEEP for the entire conversion, or the A/D accuracy may be out of specification.

12.3 Configuring Analog Port Pins

The ADCON1, TRISA and TRISE registers control the operation of the A/D port pins. The port pins that are desired as analog inputs must have their corresponding TRIS bits set (input). If the TRIS bit is cleared (output), the digital output level (VOH or VOL) will be converted.

The A/D operation is independent of the state of the CHS<2:0> bits and the TRIS bits.

Note 1: When reading the port register, all pins configured as analog input channels will read as cleared (a low level). Pins configured as digital inputs will convert an analog input. Analog levels on a digitally configured input will not affect the conversion accuracy.
2: Analog levels on any pin that is defined as a digital input, but not as an analog input, may cause the input buffer to consume current that is out of specification.
3: The TRISE register is not provided on the PIC16C745.

12.4 A/D Conversions

Note: The GO/$\overline{\text{DONE}}$ bit should **NOT** be set in the same instruction that turns on the A/D.

Clearing the GO/$\overline{\text{DONE}}$ bit during a conversion will abort the current conversion. The ADRES register will NOT be updated with the partially completed A/D conversion sample. That is, the ADRES register will continue to contain the value of the last completed conversion (or the last value written to the ADRES register). After the A/D conversion is aborted, a 2TAD wait is required before the next acquisition is started. After this 2TAD wait, an acquisition is automatically started on the selected channel.

12.5 A/D Operation During Sleep

The A/D module can operate during SLEEP mode. This requires that the A/D clock source be set to RC (ADCS<1:0> = 11). When the RC clock source is selected, the A/D module waits one instruction cycle before starting the conversion. This allows the SLEEP instruction to be executed, which eliminates all digital switching noise from the conversion. When the conversion is completed, the GO/$\overline{\text{DONE}}$ bit will be cleared, and the result loaded into the ADRES register. If the A/D interrupt is enabled, the device will wake-up from SLEEP. If the A/D interrupt is not enabled, the A/D module will then be turned off, although the ADON bit will remain set.

When the A/D clock source is another clock option (not RC), a SLEEP instruction will cause the present conversion to be aborted and the A/D module to be turned off, though the ADON bit will remain set.

Turning off the A/D places the A/D module in its lowest current consumption state.

Note: For the A/D module to operate in SLEEP, the A/D clock source must be set to RC (ADCS<1:0> = 11). To perform an A/D conversion in SLEEP, ensure the SLEEP instruction immediately follows the instruction that sets the GO/$\overline{\text{DONE}}$ bit.

12.6 Effects of a RESET

A device RESET forces all registers to their RESET state. The A/D module is disabled and any conversion in progress is aborted. All pins with analog functions are configured as available inputs.

The ADRES register will contain unknown data after a Power-on Reset.

12.7 Use of the CCP Trigger

An A/D conversion can be started by the "special event trigger" of the CCP2 module. This requires that the CCP2M<3:0> bits (CCP2CON<3:0>) be programmed as 1011 and that the A/D module is enabled (ADON bit is set). When the trigger occurs, the GO/$\overline{\text{DONE}}$ bit will be set, starting the A/D conversion, and the Timer1 counter will be reset to zero. Timer1 is reset to automatically repeat the A/D acquisition period with minimal software overhead (moving the ADRES to the desired location). The appropriate analog input channel must be selected and the minimum acquisition done before the "special event trigger" sets the GO/$\overline{\text{DONE}}$ bit (starts a conversion).

If the A/D module is not enabled (ADON is cleared), then the "special event trigger" will be ignored by the A/D module, but will still reset the Timer1 counter.

TABLE 12-2: SUMMARY OF A/D REGISTERS

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
0Bh,8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	USBIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	USBIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
1Eh	ADRES	A/D Result Register								xxxx xxxx	uuuu uuuu
1Fh	ADCON0	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	—	ADON	0000 00-0	0000 00-0
9Fh	ADCON1	—	—	—	—	—	PCFG2	PCFG1	PCFG0	---- -000	---- -000
05h	PORTA	—	—	RA5	RA4	RA3	RA2	RA1	RA0	--0x 0000	--0u 0000
85h	TRISA	—	—	PORTA Data Direction Register						--11 1111	--11 1111
09h	PORTE	—	—	—	—	—	RE2(1)	RE1(1)	RE0(1)	---- -xxx	---- -uuu
89h	TRISE	IBF(1)	OBF(1)	IBOV(1)	PSP-MODE(1)	—	PORTE(1) Data Direction Bits			0000 -111	0000 -111

Legend: x = unknown, u = unchanged, - = unimplemented read as '0'. Shaded cells are not used for A/D conversion.

Note 1: These bits are reserved on the PIC6C745; always maintain these bits clear.

NOTES:

13.0 SPECIAL FEATURES OF THE CPU

What sets a microcontroller apart from other processors are special circuits to deal with the needs of real-time applications. The PIC16C745/765 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:

- Oscillator selection
- Reset
  - Power-on Reset (POR)
  - Power-up Timer (PWRT)
  - Oscillator Start-up Timer (OST)
  - Brown-out Reset (BOR)
- Interrupts
- Watchdog Timer (WDT)
- SLEEP
- Code protection
- ID locations
- In-Circuit Serial Programming™ (ICSP)

The PIC16C745/765 has a Watchdog Timer, which can be shut off only through configuration bits. It runs off its own dedicated 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 72 ms (nominal) on power-up only and is designed to keep the part in RESET, while the power supply stabilizes. With these two timers on-chip, most applications need no external RESET circuitry.

SLEEP mode is designed to offer a very low current power-down mode. The user can wake-up from SLEEP through external RESET, WDT wake-up or through an interrupt. Several oscillator options are also made available to allow the part to fit the application. The EC oscillator allows the user to directly drive the microcontroller, while the HS oscillator allows the use of a high speed crystal/resonator. A set of configuration bits are used to select various options.

13.1 Configuration Bits

The configuration bits 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 only during programming.

REGISTER 13-1: CONFIGURATION WORD

CP1	CP0	CP1	CP0	CP1	CP0	—	—	CP1	CP0	$\overline{\text{PWRTE}}$	WDTE	FOSC1	FOSC0
bit13													bit0

Register: CONFIG
Address 2007h

bit 13-12: **CP<1:0>:** Code Protection bits(1)
11-10: 00 = All memory is code protected
9-8: 01 = Upper 3/4th of program memory code protected
5-4: 10 = Upper half of program memory code protected
11 = Code protection off

bit 7-6: **Unimplemented:** Read as '1'

bit 3: **$\overline{\text{PWRTE}}$:** Power-up Timer Enable bit
1 = PWRT disabled • No delay after Power-up Reset or Brown-out Reset
0 = PWRT enabled • A delay of 4x WDT (72 ms) is present after Power-up and Brown-out

bit 2: **WDTE:** Watchdog Timer Enable bit
1 = WDT enabled
0 = WDT disabled

bit 1-0: **FOSC<1:0>:** Oscillator Selection
00 = HS - HS osc
01 = EC - External clock. CLKOUT on OSC2 pin
10 = H4 - HS osc with 4x PLL enabled
11 = E4 - External clock with 4x PLL enabled. CLKOUT on OSC2 pin

Note 1: All of the CP<1:0> pairs have to be given the same value to enable the code protection scheme listed.

13.2 Oscillator Configurations

13.2.1 OSCILLATOR TYPES

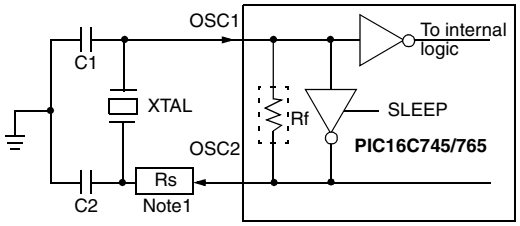
The PIC16C745/765 can be operated in four different oscillator modes. The user can program a configuration bit (FOSC0) to select one of these four modes:

- EC External Clock
- E4 External Clock with internal PLL enabled
- HS High Speed Crystal/Resonator
- H4 High Speed Crystal/Resonator with internal PLL enabled

13.2.2 CRYSTAL OSCILLATOR/CERAMIC RESONATORS

In HS mode, a crystal or ceramic resonator is connected to the OSC1/CLKIN and OSC2/CLKOUT pins to establish oscillation (Figure 13-1). The PIC16C745/765 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 HS mode, the device can have an external clock source to drive the OSC1/CLKIN pin (Figure 13-2). In this mode, the oscillator start-up timer is active for a period of 1024*TOSC. See the PIC Mid-Range MCU Reference Manual (DS33023) for details on building an external oscillator.

FIGURE 13-1: CRYSTAL/CERAMIC RESONATOR OPERATION (HS OSC CONFIGURATION)



Note 1: A series resistor may be required for AT strip cut crystals.

TABLE 13-1: CERAMIC RESONATORS

Ranges Tested:			
Mode	**Freq**	**OSC1**	**OSC2**
HS	6.0 MHz	10 - 68 pF	10 - 68 pF
These values are for design guidance only. See notes at bottom of page.			

TABLE 13-2: CAPACITOR SELECTION FOR CRYSTAL OSCILLATOR

Osc Type	Crystal Freq	Cap. Range C1	Cap. Range C2
HS	6.0 MHz	15 - 33 pF	15 - 33 pF
These values are for design guidance only. See notes at bottom of page.			

Note 1: Higher capacitance increases the stability of the oscillator, but also increases the start-up time.
2: Since each resonator/crystal has its own characteristics, the user should consult the resonator/crystal manufacturer for appropriate values of external components.
3: Rs may be required in HS mode to avoid overdriving crystals with low drive level specification.
4: When migrating from other PIC devices, oscillator performance should be verified.
5: Users should consult the USB Specification 1.1 to ensure their resonator/crystal oscillator meets the required jitter limits for USB operation.

13.2.3 H4 MODE

In H4 mode, a PLL module is switched on in-line with the clock provided across OSC1 and OCS2. The output of the PLL drives FINT.

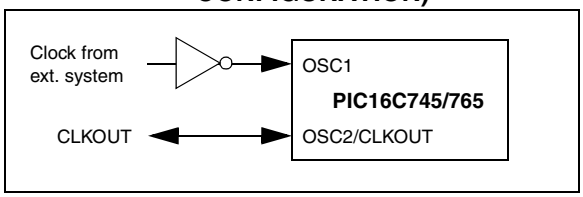
13.2.4 PLL

An on-board 4x PLL provides a cheap means of generating a stable 24 MHz FINT, using an external 6 MHz resonator. After power-up, a PLL settling time of less than TPLLRT is required.

13.2.5 EXTERNAL CLOCK IN

In EC mode, users may directly drive the PIC16C745/765 provided that this external clock source meets the AC/DC timing requirements listed in Section 17.4. Figure 13-2 below shows how an external clock circuit should be configured.

FIGURE 13-2: EXTERNAL CLOCK INPUT OPERATION (EC OSC CONFIGURATION)

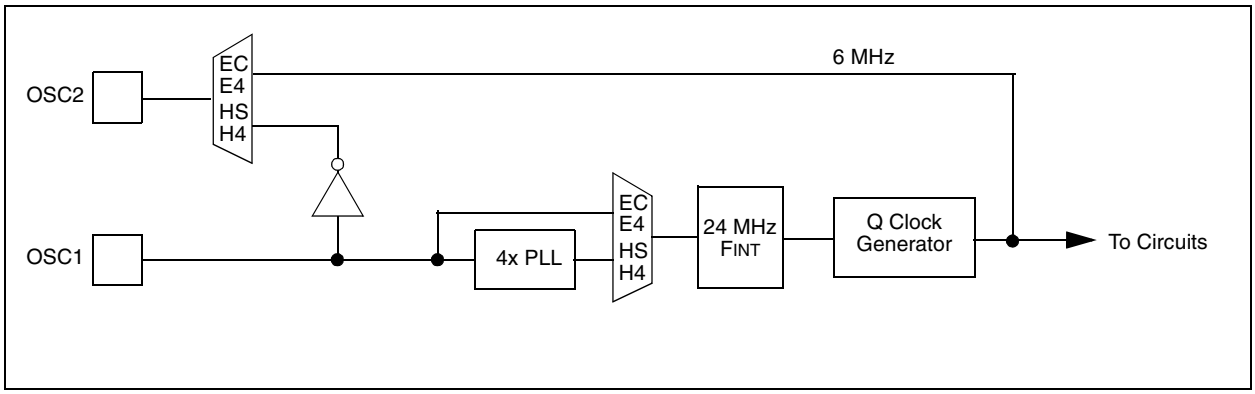


13.2.6 E4 MODE

In E4 mode, a PLL module is switched on in-line with the clock provided to OSC1. The output of the PLL drives FINT.

Note: CLKOUT is the same frequency as OSC1 if in E4 mode, otherwise CLKOUT = OSC1/4.

FIGURE 13-3: OSCILLATOR/PLL CLOCK CONTROL



13.3 RESET

The PIC16CXX differentiates between various kinds of RESET:

- Power-on Reset (POR)
- $\overline{\text{MCLR}}$ Reset during normal operation
- $\overline{\text{MCLR}}$ Reset during SLEEP
- WDT Reset (normal operation)
- Brown-out Reset (BOR)

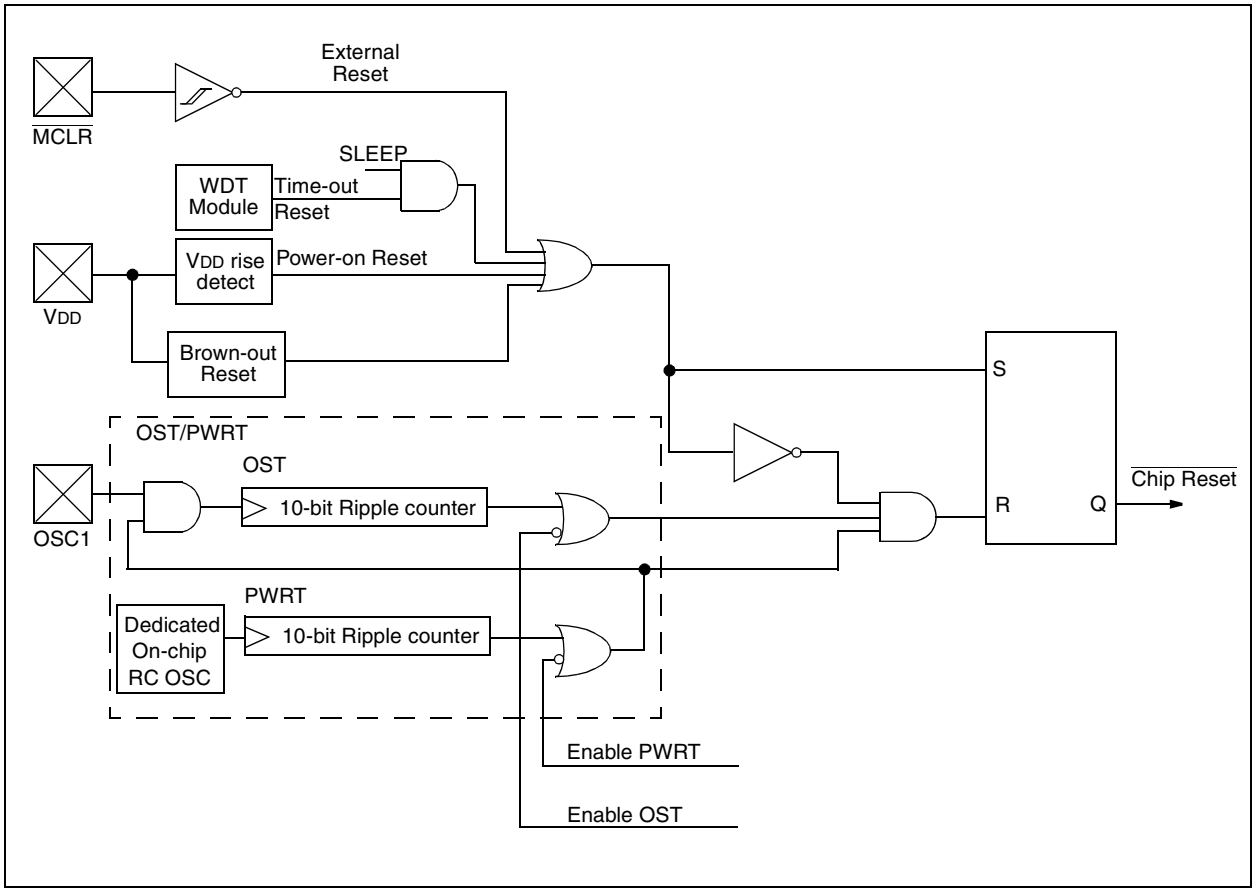
Some registers are not affected in any RESET condition; their status is unknown on POR and unchanged in any other RESET. Most other registers are reset to a "RESET state" on POR, on the $\overline{\text{MCLR}}$ and WDT Reset, on $\overline{\text{MCLR}}$ Reset during SLEEP, and on BOR. The $\overline{\text{TO}}$ and $\overline{\text{PD}}$ bits are set or cleared differently in different RESET situations as indicated in Table 13-4. These bits are used in software to determine the nature of the RESET. See Table 13-7 for a full description of RESET states of all registers.

A simplified block diagram of the on-chip RESET circuit is shown in Figure 13-4.

The PIC® devices have a $\overline{\text{MCLR}}$ noise filter in the $\overline{\text{MCLR}}$ Reset path. The filter will detect and ignore small pulses.

It should be noted that a WDT Reset does not drive $\overline{\text{MCLR}}$ pin low.

FIGURE 13-4: SIMPLIFIED BLOCK DIAGRAM OF ON-CHIP RESET CIRCUIT



13.4 RESETS

13.4.1 POWER-ON RESET (POR)

A Power-on Reset pulse is generated on-chip when VDD rise is detected (in the range of 1.5V - 2.1V). To take advantage of the POR, just tie the $\overline{MCLR}$ pin directly (or through a resistor) to VDD. This will eliminate external RC components usually needed to create a POR. A maximum rise time for VDD is specified. See Electrical Specifications for details.

When the device starts normal operation (exits the RESET condition), device operating parameters (voltage, frequency, temperature) must be met to ensure operation. If these conditions are not met, the device must be held in RESET until the operating conditions are met. Brown-out Reset may be used to meet the startup conditions.

For additional information, refer to Application Note AN607, "*Power-up Trouble Shooting*."

13.4.2 POWER-UP TIMER (PWRT)

The Power-up Timer provides a fixed 72 ms nominal time-out on power-up from the POR. The PWRT operates on an internal RC oscillator. The device is kept in RESET as long as the PWRT is active. The PWRT's time delay allows VDD to rise to an acceptable level. A configuration bit is provided to enable/disable the PWRT.

The power-up time delay will vary from chip to chip due to VDD, temperature and process variation. See DC parameters for details (TPWRT, parameter #33).

13.4.3 OSCILLATOR START-UP TIMER (OST)

The Oscillator Start-up Timer provides a delay of 1024 oscillator cycles (from OSC1 input) after the PWRT delay. This ensures that the crystal oscillator or resonator has started and stabilized.

The OST time-out is invoked only for HS mode and only on Power-on Reset or wake-up from SLEEP.

13.4.4 BROWN-OUT RESET (BOR)

If VDD falls below VBOR (parameter D005) for longer than TBOR (parameter #35), the brown-out situation will reset the device. If VDD falls below VBOR for less than TBOR, a RESET may not occur.

Once the brown-out occurs, the device will remain in Brown-out Reset until VDD rises above VBOR. The Power-up Timer then keeps the device in RESET for TPWRT (parameter #33). If VDD should fall below VBOR during TPWRT, the Brown-out Reset process will restart when VDD rises above VBOR, with the Power-up Timer Reset. Since the device is intended to operate at 5V nominal only, the Brown-out Detect is always enabled and the device will RESET when Vdd falls below the brown-out threshold. This device is unique in that the 4•WDT timer will not activate after a brown-out if $\overline{PWRTE}$ = 1 (inactive).

13.4.5 TIME-OUT SEQUENCE

On power-up, the time-out sequence is as follows: The PWRT delay starts (if enabled), when a Power-on Reset occurs. Then OST starts counting 1024 oscillator cycles when PWRT ends (HS). When the OST ends, the device comes out of RESET.

If $\overline{MCLR}$ is kept low long enough, the time-outs will expire. Bringing $\overline{MCLR}$ high will begin execution immediately. This is useful for testing purposes or to synchronize more than one PIC16CXX device operating in parallel.

Table 13-5 shows the RESET conditions for the STATUS, PCON and PC registers, while Table 13-7 shows the RESET conditions for all the registers.

13.4.6 POWER CONTROL/STATUS REGISTER (PCON)

The Brown-out Reset Status bit, $\overline{BOR}$, is unknown on a POR. It must be set by the user and checked on subsequent RESETS to see if bit $\overline{BOR}$ was cleared, indicating a BOR occurred. The $\overline{BOR}$ bit is not predictable if the Brown-out Reset circuitry is disabled.

The Power-on Reset Status bit, $\overline{POR}$, is cleared on a POR and unaffected otherwise. The user must set this bit following a POR and check it on subsequent RESETS to see if it has been cleared.

13.5 Time-out in Various Situations

TABLE 13-3: RESET TIME-OUTS

Oscillator Configuration	POR		BOR		Wake-up from SLEEP
	$\overline{PWRTE}$ = 0	$\overline{PWRTE}$ = 1	$\overline{PWRTE}$ = 0	$\overline{PWRTE}$ = 1	
HS	TPWRT + 1024•TOSC	1024•TOSC	TPWRT + 1024•TOSC	1024•TOSC	1024•TOSC
H4	TPWRT + TPLLRT + 1024•TOSC	TPLLRT + 1024•TOSC	TPWRT + TPLLRT + 1024•TOSC	TPLLRT + 1024•TOSC	TPLLRT + 1024•TOSC
EC	TPWRT	0	TPWRT	0	0
E4	TPWRT + TPLLRT	TPLLRT	TPWRT + TPLLRT	TPLLRT	TPLLRT

TABLE 13-4: STATUS BITS AND THEIR SIGNIFICANCE

$\overline{POR}$	$\overline{BOR}$	$\overline{TO}$	$\overline{PD}$	
0	x	1	1	Power-on Reset
0	x	0	x	Illegal, $\overline{TO}$ is set on $\overline{POR}$
0	x	x	0	Illegal, $\overline{PD}$ is set on $\overline{POR}$
1	0	1	1	Brown-out Reset
1	1	0	1	WDT Reset
1	1	0	0	WDT Wake-up
1	1	u	u	$\overline{MCLR}$ Reset during normal operation
1	1	1	0	$\overline{MCLR}$ Reset during SLEEP or Interrupt Wake-up from SLEEP

TABLE 13-5: RESET CONDITION FOR SPECIAL REGISTERS

Condition	Program Counter	STATUS Register	PCON Register
Power-on Reset	000h	0001 1xxx	---- --0x
$\overline{MCLR}$ Reset during normal operation	000h	000u uuuu	---- --uu
$\overline{MCLR}$ Reset during SLEEP	000h	0001 0uuu	---- --uu
WDT Reset	000h	0000 1uuu	---- --uu
WDT Wake-up	PC + 1	uuu0 0uuu	---- --uu
Brown-out Reset	000h	000x xuuu	---- --u0
Interrupt Wake-up from SLEEP	PC + 1(1)	uuu1 0uuu	---- --uu

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

Note 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 13-6: REGISTERS ASSOCIATED WITH RESETS

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
03h, 83h, 103h, 183h	Status	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	000q quuu
8Eh	PCON	—	—	—	—	—	—	$\overline{POR}$	$\overline{BOR}$	---- --qq	---- --uu

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

TABLE 13-7: INITIALIZATION CONDITIONS FOR ALL REGISTERS

Register	Power-on Reset Brown-out Reset	$\overline{\text{MCLR}}$ Resets WDT Reset	Wake-up via WDT or Interrupt
W	xxxx xxxx	uuuu uuuu	uuuu uuuu
INDF	N/A	N/A	N/A
TMR0	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCL	0000h	0000h	PC + 1(2)
STATUS	0001 1xxx	000q quuu(3)	uuuq quuu(3)
FSR	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTA	--0x 0000	--0u 0000	--uu uuuu
PORTB	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTC	xx-- -xxx	uu-- -uuu	uu-- -uuu
PORTD(4)	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTE(4)	---- -xxx	---- -uuu	---- -uuu
PCLATH	---0 0000	---0 0000	---u uuuu
INTCON	0000 000x	0000 000u	uuuu uuuu(1)
PIR1	0000 0000	0000 0000	uuuu uuuu(1)
PIR2	---- ---0	---- ---0	---- ---u(1)
TMR1L	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR1H	xxxx xxxx	uuuu uuuu	uuuu uuuu
T1CON	--00 0000	--uu uuuu	--uu uuuu
TMR2	0000 0000	0000 0000	uuuu uuuu
T2CON	-000 0000	-000 0000	-uuu uuuu
CCPR1L	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR1H	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCP1CON	--00 0000	--00 0000	--uu uuuu
RCSTA	0000 -00x	0000 -00x	uuuu -uuu
TXREG	0000 0000	0000 0000	uuuu uuuu
RCREG	0000 0000	0000 0000	uuuu uuuu
CCPR2L	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR2H	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCP2CON	0000 0000	0000 0000	uuuu uuuu
ADRES	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADCON0	0000 00-0	0000 00-0	uuuu uu-u
OPTION_REG	1111 1111	1111 1111	uuuu uuuu
TRISA	--11 1111	--11 1111	--uu uuuu
TRISB	1111 1111	1111 1111	uuuu uuuu
TRISC	11-- -111	11-- -111	uu-- -uuu

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

Note 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 13-5 for RESET value for specific condition.

4: PIC16C765 only.

TABLE 13-7: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Power-on Reset Brown-out Reset	$\overline{\text{MCLR}}$ Resets WDT Reset	Wake-up via WDT or Interrupt
TRISD(4)	1111 1111	1111 1111	uuuu uuuu
TRISE(4)	0000 -111	0000 -111	uuuu -uuu
PIE1	0000 0000	0000 0000	uuuu uuuu
PIE2	---- ---0	---- ---0	---- ---u
PCON	---- --0q(3)	---- --uu	---- --uu
PR2	1111 1111	1111 1111	1111 1111
TXSTA	0000 -010	0000 -010	uuuu -uuu
SPBRG	0000 0000	0000 0000	uuuu uuuu
ADCON1	---- -000	---- -000	---- -uuu
UIR	--00 0000	--00 0000	--00 0000
UIE	--00 0000	--00 0000	--00 0000
UEIR	0000 0000	0000 0000	0000 0000
UEIE	0000 0000	0000 0000	0000 0000
USTAT	---x xx--	---u uu--	---u uu--
UCTRL	--x0 000-	--xq qqq-	--xq qqq-
UADDR	-000 0000	-000 0000	-000 0000
USWSTAT	0000 0000	0000 0000	0000 0000
UEP0	---- 0000	---- 0000	---- 0000
UEP1	---- 0000	---- 0000	---- 0000
UEP2	---- 0000	---- 0000	---- 0000

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

Note 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 13-5 for RESET value for specific condition.

4: PIC16C765 only.

13.6 Interrupts

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

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

A global interrupt enable bit, GIE (INTCON<7>) enables (if set) all unmasked interrupts or disables (if cleared) all interrupts. When bit GIE is enabled, and an interrupt's flag bit and mask bit are set, the interrupt will vector immediately. Individual interrupts can be disabled through their corresponding enable bits in various registers. Individual interrupt bits are set, regardless of the status of the GIE bit. The GIE bit is cleared on RESET.

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

The RB0/INT pin interrupt, the RB port change interrupt and the TMR0 overflow interrupt flags are contained in the INTCON register.

The peripheral interrupt flags are contained in the special function registers PIR1 and PIR2. The corresponding interrupt enable bits are contained in special function registers PIE1 and PIE2 and the peripheral interrupt enable bit is contained in special function register INTCON.

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

For external interrupt events, such as the INT pin or PORTB change interrupt, the interrupt latency will be three or four instruction cycles. The exact latency depends when the interrupt event occurs. The latency is the same for one or two cycle instructions. Individual interrupt flag bits are set, regardless of the status of their corresponding mask bit or the GIE bit.

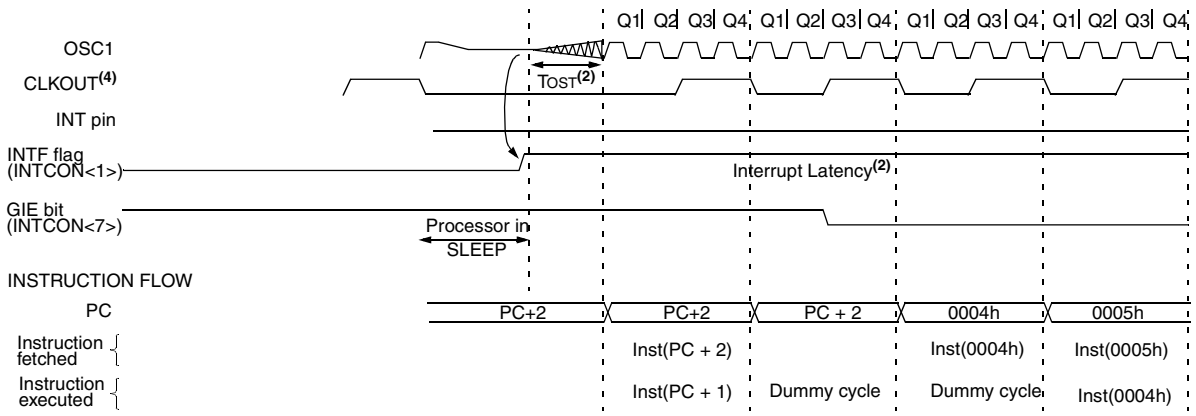
Note: If an interrupt occurs while the Global Interrupt Enable (GIE) bit is being cleared, the GIE bit may unintentionally be re-enabled by the user's Interrupt Service Routine (the `RETFIE` instruction). The events that would cause this to occur are:

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 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.

Perform the following to ensure that interrupts are globally disabled:

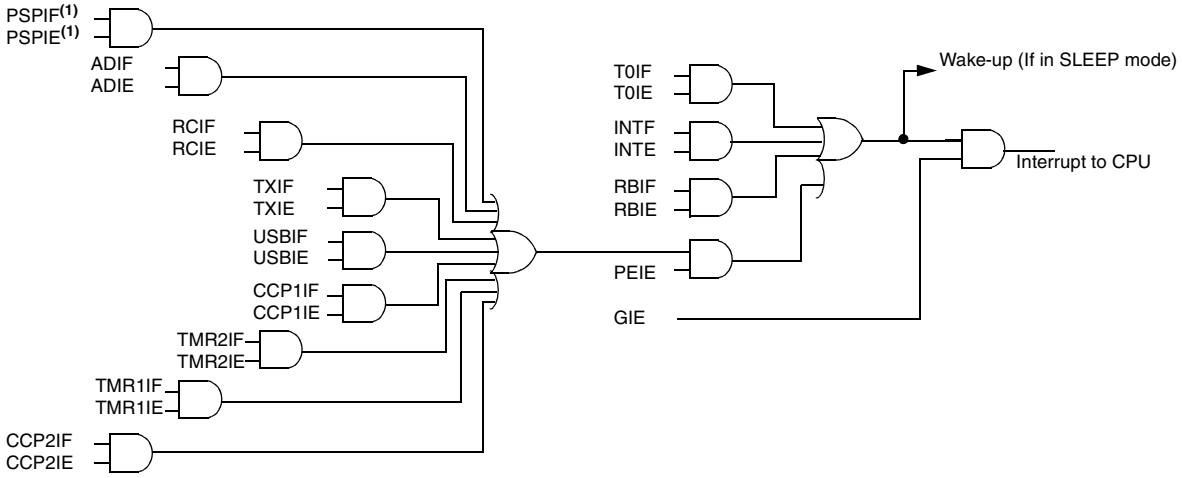
```
LOOP BCF   INTCON, GIE  ; Disable global
                        ;   interrupt bit
     BTFSC INTCON, GIE  ; Global interrupt
                        ;   disabled?
     GOTO  LOOP         ; NO, try again
        :               ;   Yes, continue
                        ;   with program
                        ;   flow
```

FIGURE 13-5: WAKE-UP FROM SLEEP THROUGH INTERRUPT



Note 1: HS oscillator mode assumed.
2: TOST = 1024TOSC (drawing not to scale). This delay is not present in EC osc mode.
3: GIE = '1' assumed. After wake-up, the processor jumps to the 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.

FIGURE 13-6: INTERRUPT LOGIC



The following table shows the interrupts for each device.

Device	T0IF	INTF	RBIF	PSPIF	ADIF	RCIF	TXIF	USBIF	CCP1IF	TMR2IF	TMR1IF	CCP2IF
PIC16C745	Yes	Yes	Yes	—	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
PIC16C765	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Note 1: PIC16C765 only.

13.6.1 INT INTERRUPT

The external interrupt on RB0/INT pin is edge triggered: either rising, if bit INTEDG (OPTION_REG<6>) is set, or falling, if the INTEDG bit is clear. When a valid edge appears on the RB0/INT pin, flag bit INTF (INTCON<1>) is set. This interrupt can be disabled by clearing enable bit INTE (INTCON<4>). Flag bit INTF 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 bit INTE was set prior to going into SLEEP. The status of global interrupt enable bit GIE, decides whether or not the processor branches to the interrupt vector following wake-up. See Section 13.9 for details on SLEEP mode.

13.6.2 TMR0 INTERRUPT

An overflow (FFh → 00h) in the TMR0 register will set flag bit T0IF (INTCON<2>). The interrupt can be enabled/disabled by setting/clearing enable bit T0IE (INTCON<5>). (Section 6.0)

13.6.3 PORTB INTERRUPT ON CHANGE

An input change on PORTB<7:4> sets flag bit RBIF (INTCON<0>). The interrupt can be enabled/disabled by setting/clearing enable bit RBIE (INTCON<3>) (Section 5.2).

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 get set.

13.7 Context Saving During Interrupts

During an interrupt, only the PC is saved on the stack. At the very least, W and STATUS should be saved to preserve the context for the interrupted program. All registers that may be corrupted by the ISR, such as PCLATH or FSR, should be saved.

Example 13-1 stores and restores the STATUS, W and PCLATH registers. The register, W_TEMP, is defined in Common RAM, the last 16 bytes of each bank that may be accessed from any bank. The STATUS_TEMP and PCLATH_TEMP are defined in bank 0.

The example:

a) Stores the W register.
b) Stores the STATUS register in bank 0.
c) Stores the PCLATH register in bank 0.
d) Executes the ISR code.
e) Restores the PCLATH register.
f) Restores the STATUS register
g) Restores W.

Note that W_TEMP, STATUS_TEMP and PCLATH_TEMP are defined in the common RAM area (70h - 7Fh) to avoid register bank switching during context save and restore.

EXAMPLE 13-1: SAVING STATUS, W, AND PCLATH REGISTERS IN RAM

```
#define     W_TEMP              0x70
#define     STATUS_TEMP         0x71
#define     PCLATH_TEMP         0x72
    org     0x04                ; start at Interrupt Vector
    MOVWF    W_TEMP             ; Save W register
    MOVF    STATUS,W
    MOVWF   STATUS_TEMP         ; save STATUS
    MOVF    PCLATH,W
    MOVWF   PCLATH_TEMP         ; save PCLATH
    :
    (Interrupt Service Routine)
    :
    MOVF    PCLATH_TEMP,W
    MOVWF   PCLATH
    MOVF    STATUS_TEMP,W
    MOVWF   STATUS
    SWAPF   W_TEMP,F            ;
    SWAPF   W_TEMP,W            ; swapf loads W without affecting STATUS flags
    RETFIE
```

13.8 Watchdog Timer (WDT)

The Watchdog Timer is a free running on-chip dedicated oscillator, which does not require any external components. The WDT will run, even if the clock on the OSC1/CLKIN and OSC2/CLKOUT pins of the device has been stopped, for example, by execution of a SLEEP instruction.

During normal operation, a WDT time-out generates a device RESET (Watchdog Timer Reset). If the device is in SLEEP mode, a WDT time-out causes the device to wake-up and resume normal operation (Watchdog Timer Wake-up).

The WDT can be permanently disabled by clearing configuration bit WDTE (Section 13.1).

13.8.1 WDT PERIOD

The WDT has a nominal time-out period of 18 ms (parameter #31, TWDT). The time-out periods vary with temperature, VDD and process variations. 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. Time-out periods up to 128 TWDT can be realized.

The CLRWDT and SLEEP instructions clear the WDT and the postscaler, if assigned to the WDT. In addition, the SLEEP instruction prevents the WDT from generating a RESET, but will allow the WDT to wake the device from SLEEP mode.

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

13.8.2 WDT PROGRAMMING CONSIDERATIONS

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

Note:	When a CLRWDT instruction is executed and the prescaler is assigned to the WDT, the prescaler count will be cleared, but the prescaler assignment is not changed.

FIGURE 13-7: WATCHDOG TIMER BLOCK DIAGRAM

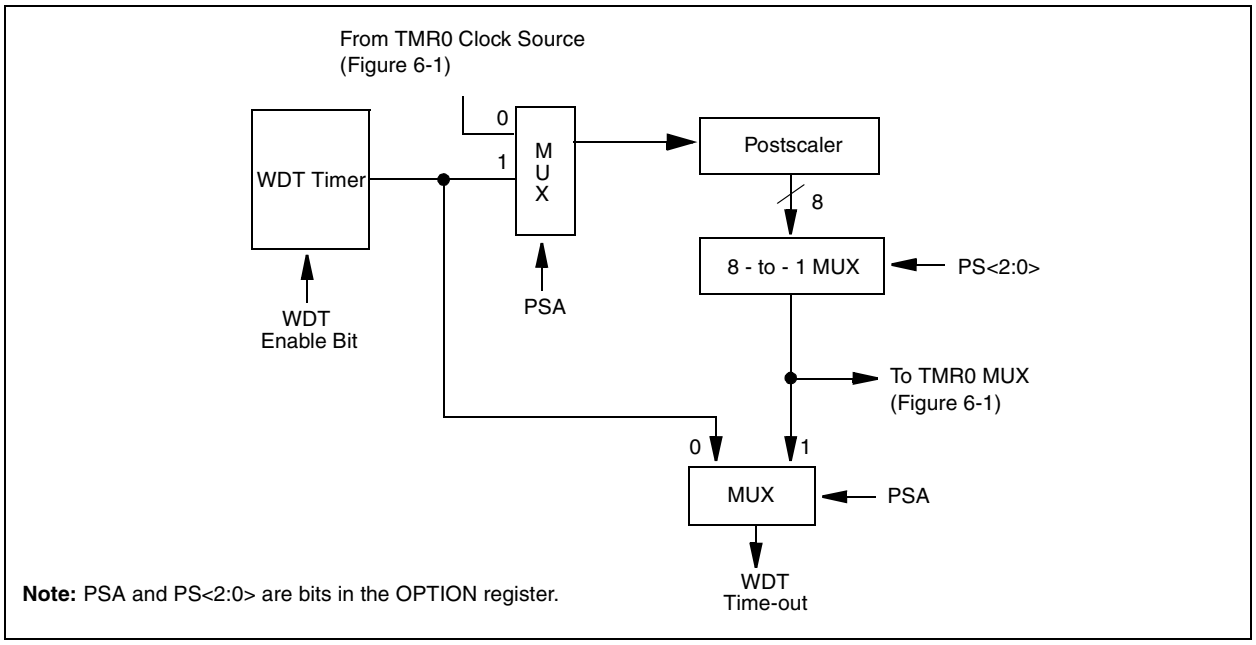


TABLE 13-8: SUMMARY OF WATCHDOG TIMER REGISTERS

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on All Other Resets
2007h	Config. bits	—	BODEN(1)	CP1	CP0	PWRTE(1)	WDTE	PLL	FOSC0		
81h,181h	OPTION_REG	$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

Legend: Shaded cells are not used by the Watchdog Timer.
Note 1: See Register 13-1 for operation of these bits.

13.9 Power-Down Mode (SLEEP)

Power-down mode is entered by executing a SLEEP instruction.

If enabled, the WDT will be cleared but keeps running, the $\overline{PD}$ bit (STATUS<3>) is cleared, the $\overline{TO}$ (STATUS<4>) bit is set, and the oscillator driver is turned off. The I/O ports maintain the status they had, before the SLEEP instruction was executed (driving high, low, or hi-impedance).

For lowest current consumption in this mode, place all I/O pins at either VDD or VSS, ensure no external circuitry is drawing current from the I/O pin, power-down the A/D, and disable external clocks. Pull all I/O pins that are hi-impedance inputs, high or low externally, to avoid switching currents caused by floating inputs. The T0CKI 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).

13.9.1 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 Wake-up (if WDT was enabled).
3. Interrupt from INT pin, RB port change or some Peripheral Interrupts.

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

The following peripheral interrupts can wake the device from SLEEP:

1. TMR1 interrupt. Timer1 must be operating as an asynchronous counter.
2. USB interrupt.
3. CCP capture mode interrupt.
4. Parallel slave port read or write (PIC16C765 only).
5. A/D conversion (when A/D clock source is dedicated internal oscillator).
6. USART TX or RX (Synchronous Slave mode).

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

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 then 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.

13.9.2 WAKE-UP USING INTERRUPTS

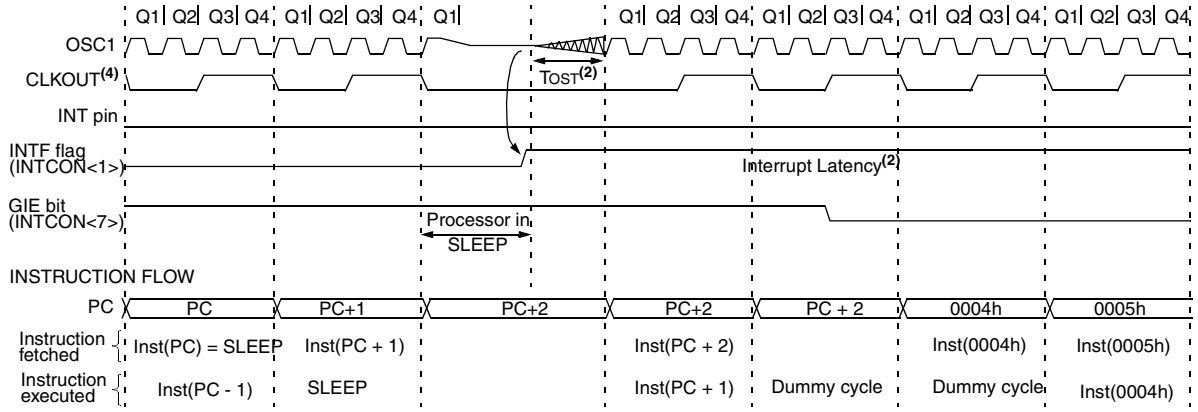
When global interrupts are disabled (GIE cleared) and any interrupt source has both its interrupt enable bit and interrupt flag bit set, one of the following will occur:

- If the interrupt occurs **before** the execution of a SLEEP instruction, the SLEEP instruction will complete as a NOP. Therefore, the WDT and WDT postscaler will not be cleared, the $\overline{TO}$ bit will not be set and $\overline{PD}$ bit will not be cleared.
- If the interrupt occurs **during or after** the execution of a SLEEP instruction, the device will immediately wake up from sleep. The SLEEP instruction will be completely executed before the wake-up. Therefore, the WDT and WDT postscaler will be cleared, the $\overline{TO}$ bit will be set and the $\overline{PD}$ bit will be cleared.

Even if the flag bits were checked before executing a SLEEP instruction, it may be possible for flag bits to become set before the SLEEP instruction completes. To determine whether a SLEEP instruction executed, test the $\overline{PD}$ bit. If the $\overline{PD}$ bit is set, the SLEEP instruction was executed as a NOP.

To ensure that the WDT is cleared, a CLRWDT instruction should be executed before a SLEEP instruction.

FIGURE 13-8: WAKE-UP FROM SLEEP THROUGH INTERRUPT



Note 1: HS oscillator mode assumed.
2: TOST = 1024TOSC (drawing not to scale). This delay is not present in EC osc mode.
3: GIE = '1' assumed. After wake-up, the processor jumps to the 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.

13.10 Program Verification/Code Protection

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

Note:	Microchip does not recommend code protecting windowed devices. Devices that are code protected may be erased, but not programmed again.

13.11 ID Locations

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

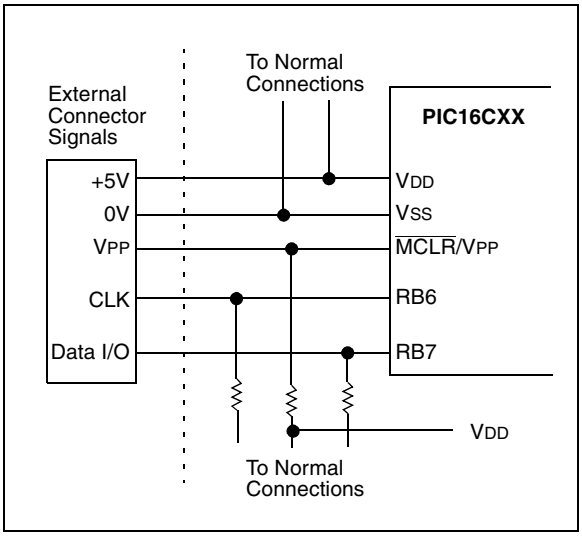
13.12 In-Circuit Serial Programming

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

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

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

FIGURE 13-9: TYPICAL IN-CIRCUIT SERIAL PROGRAMMING CONNECTION



14.0 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 14-2 lists **byte-oriented**, **bit-oriented**, and **literal and control** operations. Table 14-1 shows the opcode field descriptions.

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

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

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

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

TABLE 14-1: OPCODE FIELD DESCRIPTIONS

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

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

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

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

Table 14-2 lists the instructions recognized by the MPASM assembler.

Figure 14-1 shows the general formats that the instructions can have.

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

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

0xhh

where h signifies a hexadecimal digit.

FIGURE 14-1: GENERAL FORMAT FOR INSTRUCTIONS

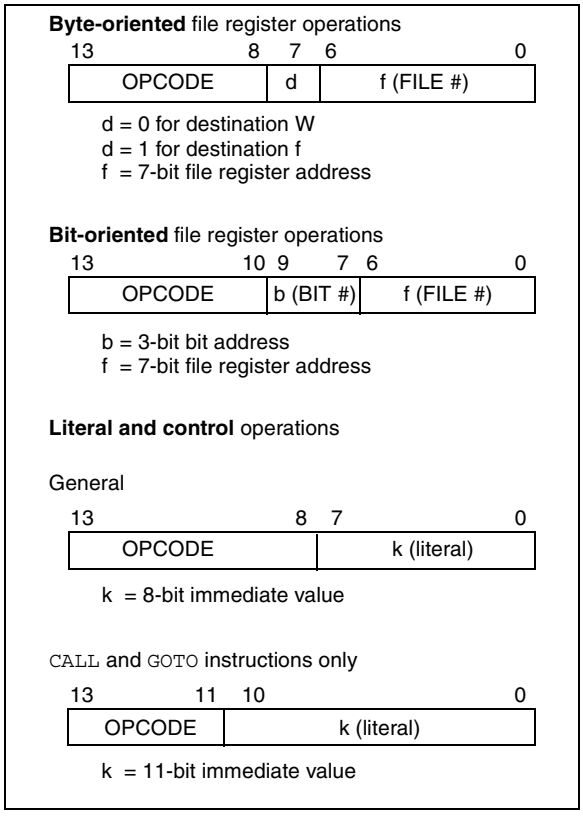


TABLE 14-2: PIC16CXX INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb	LSb	Status Affected	Notes
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 with f	1	00 0101 dfff ffff		Z	**1,2**
CLRF	**f**	Clear f	1	00 0001 lfff ffff		Z	**2**
CLRW	**-**	Clear W	1	00 0001 0000 0011		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 with 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 lfff 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 nibbles in f	1	00 1110 dfff ffff			**1,2**
XORWF	**f, d**	Exclusive OR W with 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 and W	1	11 111x kkkk kkkk		C,DC,Z	
ANDLW	**k**	AND literal with 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 with 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**	Exclusive OR literal with W	1	11 1010 kkkk kkkk		Z	

Note 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 Timer0 Module.

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`.

Note: Additional information on the mid-range instruction set is available in the PIC Mid-Range MCU Family Reference Manual (DS33023).

14.1 Instruction Descriptions

ADDLW	Add Literal and W
Syntax:	[*label*] ADDLW k
Operands:	$0 \le k \le 255$
Operation:	(W) + k → (W)
Status Affected:	C, DC, Z
Description:	The contents of the W register are added to the eight bit literal 'k' and the result is placed in the W register.

ADDWF	Add W and f
Syntax:	[*label*] ADDWF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(W) + (f) → (destination)
Status Affected:	C, DC, Z
Description:	Add the contents of the W register with register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'.

ANDLW	AND Literal with W
Syntax:	[*label*] ANDLW k
Operands:	$0 \le k \le 255$
Operation:	(W) .AND. (k) → (W)
Status Affected:	Z
Description:	The contents of W register are AND'ed with the eight bit literal 'k'. The result is placed in the W register.

ANDWF	AND W with f
Syntax:	[*label*] ANDWF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(W) .AND. (f) → (destination)
Status Affected:	Z
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'.

BCF	Bit Clear f
Syntax:	[*label*] BCF f,b
Operands:	$0 \le f \le 127$ $0 \le b \le 7$
Operation:	0 → (f<b>)
Status Affected:	None
Description:	Bit 'b' in register 'f' is cleared.

BSF	Bit Set f
Syntax:	[*label*] BSF f,b
Operands:	$0 \le f \le 127$ $0 \le b \le 7$
Operation:	1 → (f<b>)
Status Affected:	None
Description:	Bit 'b' in register 'f' is set.

BTFSS	Bit Test f, Skip if Set
Syntax:	[*label*] BTFSS f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b < 7
Operation:	skip if (f<b>) = 1
Status Affected:	None
Description:	If bit 'b' in register 'f' is '0', the next instruction is executed. If bit 'b' is '1', then the next instruction is discarded and a NOP is executed instead making this a 2TCY instruction.

BTFSC	Bit Test, Skip if Clear
Syntax:	[*label*] BTFSC f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operation:	skip if (f<b>) = 0
Status Affected:	None
Description:	If bit 'b' in register 'f' is '1', the next instruction is executed. If bit 'b', in register 'f', is '0', the next instruction is discarded, and a NOP is executed instead, making this a 2TCY instruction.

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

CLRF	Clear f
Syntax:	[*label*] CLRF f
Operands:	0 ≤ f ≤ 127
Operation:	00h → (f) 1 → Z
Status Affected:	Z
Description:	The contents of register 'f' are cleared and the Z bit is set.

CLRW	Clear W
Syntax:	[*label*] CLRW
Operands:	None
Operation:	00h → (W) 1 → Z
Status Affected:	Z
Description:	W register is cleared. Zero bit (Z) is set.

CLRWDT	Clear Watchdog Timer
Syntax:	[*label*] CLRWDT
Operands:	None
Operation:	00h → WDT 0 → WDT prescaler, 1 → $\overline{TO}$ 1 → $\overline{PD}$
Status Affected:	$\overline{TO}$, $\overline{PD}$
Description:	CLRWDT instruction resets the Watchdog Timer. It also resets the prescaler of the WDT. Status bits $\overline{TO}$ and $\overline{PD}$ are set.

COMF	Complement f
Syntax:	[*label*] COMF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	$(\bar{f}) \rightarrow$ (destination)
Status Affected:	Z
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'.

GOTO	Unconditional Branch
Syntax:	[*label*] GOTO k
Operands:	$0 \le k \le 2047$
Operation:	$k \rightarrow$ PC<10:0> PCLATH<4:3> $\rightarrow$ PC<12:11>
Status Affected:	None
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.

DECF	Decrement f
Syntax:	[*label*] DECF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(f) - 1 $\rightarrow$ (destination)
Status Affected:	Z
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'.

INCF	Increment f
Syntax:	[*label*] INCF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(f) + 1 $\rightarrow$ (destination)
Status Affected:	Z
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'.

DECFSZ	Decrement f, Skip if 0
Syntax:	[*label*] DECFSZ f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(f) - 1 $\rightarrow$ (destination); skip if result = 0
Status Affected:	None
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 1, the next instruction is executed. If the result is 0, then a NOP is executed instead making it a 2TCY instruction.

INCFSZ	Increment f, Skip if 0
Syntax:	[*label*] INCFSZ f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(f) + 1 $\rightarrow$ (destination), skip if result = 0
Status Affected:	None
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 1, the next instruction is executed. If the result is 0, a NOP is executed instead making it a 2TCY instruction.

IORLW	Inclusive OR Literal with W
Syntax:	[*label*] IORLW k
Operands:	$0 \le k \le 255$
Operation:	(W) .OR. k → (W)
Status Affected:	Z
Description:	The contents of the W register are OR'ed with the eight bit literal 'k'. The result is placed in the W register.

IORWF	Inclusive OR W with f
Syntax:	[*label*] IORWF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(W) .OR. (f) → (destination)
Status Affected:	Z
Description:	Inclusive OR the W register with register 'f'. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.

MOVF	Move f
Syntax:	[*label*] MOVF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(f) → (destination)
Status Affected:	Z
Description:	The contents of register f are moved to a destination dependent upon the status of 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.

MOVLW	Move Literal to W
Syntax:	[*label*] MOVLW k
Operands:	$0 \le k \le 255$
Operation:	k → (W)
Status Affected:	None
Description:	The eight bit literal 'k' is loaded into W register. The don't cares will assemble as 0's.

MOVWF	Move W to f
Syntax:	[*label*] MOVWF f
Operands:	$0 \le f \le 127$
Operation:	(W) → (f)
Status Affected:	None
Description:	Move data from W register to register 'f'.

NOP	No Operation
Syntax:	[*label*] NOP
Operands:	None
Operation:	No operation
Status Affected:	None
Description:	No operation.

RETFIE **Return from Interrupt**

Syntax: [*label*] RETFIE

Operands: None

Operation: TOS → PC,
1 → GIE

Status Affected: None

RETLW **Return with Literal in W**

Syntax: [*label*] RETLW k

Operands: 0 ≤ k ≤ 255

Operation: k → (W);
TOS → PC

Status Affected: None

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.

RETURN **Return from Subroutine**

Syntax: [*label*] RETURN

Operands: None

Operation: TOS → PC

Status Affected: None

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

RLF **Rotate Left f through Carry**

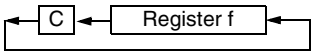
Syntax: [*label*] RLF f,d

Operands: 0 ≤ f ≤ 127
d ∈ [0,1]

Operation: See description below

Status Affected: C

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



RRF **Rotate Right f through Carry**

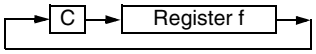
Syntax: [*label*] RRF f,d

Operands: 0 ≤ f ≤ 127
d ∈ [0,1]

Operation: See description below

Status Affected: C

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'.



SLEEP

Syntax: [*label*] SLEEP

Operands: None

Operation: 00h → WDT,
0 → WDT prescaler,
1 → $\overline{TO}$,
0 → $\overline{PD}$

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

Description: The power-down status bit, $\overline{PD}$ is cleared. Time-out status bit, $\overline{TO}$ is set. Watchdog Timer and its prescaler are cleared.
The processor is put into SLEEP mode with the oscillator stopped. See Section 13.9 for more details.

SUBLW **Subtract W from Literal**

Syntax: [*label*] SUBLW k

Operands: $0 \le k \le 255$

Operation: k - (W) → (W)

Status Affected: C, DC, Z

Description: The W register is subtracted (2's complement method) from the eight bit literal 'k'. The result is placed in the W register.

SUBWF **Subtract W from f**

Syntax: [*label*] SUBWF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (f) - (W) → (destination)

Status Affected: C, DC, Z

Description: Subtract (2's complement method) 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'.

SWAPF **Swap Nibbles in f**

Syntax: [*label*] SWAPF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (f<3:0>) → (destination<7:4>),
(f<7:4>) → (destination<3:0>)

Status Affected: None

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'.

XORLW **Exclusive OR Literal with W**

Syntax: [*label*] XORLW k

Operands: $0 \le k \le 255$

Operation: (W) .XOR. k → (W)

Status Affected: Z

Description: The contents of the W register are XOR'ed with the eight bit literal 'k'. The result is placed in the W register.

XORWF **Exclusive OR W with f**

Syntax: [*label*] XORWF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (W) .XOR. (f) → (destination)

Status Affected: Z

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'.

15.0 DEVELOPMENT SUPPORT

The PIC® microcontrollers are supported with a full range of hardware and software development tools:

- Integrated Development Environment
  - MPLAB® IDE Software
- Assemblers/Compilers/Linkers
  - MPASM Assembler
  - MPLAB-C17 and MPLAB-C18 C Compilers
  - MPLINK/MPLIB Linker/Librarian
- Simulators
  - MPLAB-SIM Software Simulator
- Emulators
  - MPLAB-ICE Real-Time In-Circuit Emulator
  - ICEPIC™
- In-Circuit Debugger
  - MPLAB-ICD for PIC16F87X
- Device Programmers
  - PRO MATE® II Universal Programmer
  - PICSTART® Plus Entry-Level Prototype Programmer
- Low-Cost Demonstration Boards
  - PICDEM-1
  - PICDEM-2
  - PICDEM-3
  - PICDEM-17
  - KEELOQ®

15.1 MPLAB Integrated Development Environment Software

The MPLAB IDE software brings an ease of software development previously unseen in the 8-bit microcontroller market. MPLAB is a Windows®-based application which contains:

- Multiple functionality
  - editor
  - simulator
  - programmer (sold separately)
  - emulator (sold separately)
- A full featured editor
- A project manager
- Customizable tool bar and key mapping
- A status bar
- On-line help

MPLAB allows you to:

- Edit your source files (either assembly or 'C')
- One touch assemble (or compile) and download to PIC MCU tools (automatically updates all project information)
- Debug using:
  - source files
  - absolute listing file
  - object code

The ability to use MPLAB with Microchip's simulator, MPLAB-SIM, allows a consistent platform and the ability to easily switch from the cost-effective simulator to the full featured emulator with minimal retraining.

15.2 MPASM Assembler

MPASM is a full featured universal macro assembler for all PIC MCUs. It can produce absolute code directly in the form of HEX files for device programmers, or it can generate relocatable objects for MPLINK.

MPASM has a command line interface and a Windows shell and can be used as a stand-alone application on a Windows 3.x or greater system. MPASM generates relocatable object files, Intel standard HEX files, MAP files to detail memory usage and symbol reference, an absolute LST file which contains source lines and generated machine code, and a COD file for MPLAB debugging.

MPASM features include:

- MPASM and MPLINK are integrated into MPLAB projects.
- MPASM allows user defined macros to be created for streamlined assembly.
- MPASM allows conditional assembly for multi purpose source files.
- MPASM directives allow complete control over the assembly process.

15.3 MPLAB-C17 and MPLAB-C18 C Compilers

The MPLAB-C17 and MPLAB-C18 Code Development Systems are complete ANSI 'C' compilers and integrated development environments for Microchip's PIC17CXXX and PIC18CXXX family of microcontrollers, respectively. These compilers provide powerful integration capabilities and ease of use not found with other compilers.

For easier source level debugging, the compilers provide symbol information that is compatible with the MPLAB IDE memory display.

15.4 MPLINK/MPLIB Linker/Librarian

MPLINK is a relocatable linker for MPASM and MPLAB-C17 and MPLAB-C18. It can link relocatable objects from assembly or C source files along with pre-compiled libraries using directives from a linker script.

MPLIB is a librarian for pre-compiled code to be used with MPLINK. When a routine from a library is called from another source file, only the modules that contains that routine will be linked in with the application. This allows large libraries to be used efficiently in many different applications. MPLIB manages the creation and modification of library files.

MPLINK features include:

- MPLINK works with MPASM and MPLAB-C17 and MPLAB-C18.
- MPLINK allows all memory areas to be defined as sections to provide link-time flexibility.

MPLIB features include:

- MPLIB makes linking easier because single libraries can be included instead of many smaller files.
- MPLIB helps keep code maintainable by grouping related modules together.
- MPLIB commands allow libraries to be created and modules to be added, listed, replaced, deleted, or extracted.

15.5 MPLAB-SIM Software Simulator

The MPLAB-SIM Software Simulator allows code development in a PC host environment by simulating the PIC series microcontrollers on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a file or user-defined key press to any of the pins. The execution can be performed in single step, execute until break, or trace mode.

MPLAB-SIM fully supports symbolic debugging using MPLAB-C17 and MPLAB-C18 and MPASM. The Software Simulator offers the flexibility to develop and debug code outside of the laboratory environment making it an excellent multi-project software development tool.

15.6 MPLAB-ICE High Performance Universal In-Circuit Emulator with MPLAB IDE

The MPLAB-ICE Universal In-Circuit Emulator is intended to provide the product development engineer with a complete microcontroller design tool set for PIC microcontrollers (MCUs). Software control of MPLAB-ICE is provided by the MPLAB Integrated Development Environment (IDE), which allows editing, “make” and download, and source debugging from a single environment.

Interchangeable processor modules allow the system to be easily reconfigured for emulation of different processors. The universal architecture of the MPLAB-ICE allows expansion to support new PIC microcontrollers.

The MPLAB-ICE Emulator System has been designed as a real-time emulation system with advanced features that are generally found on more expensive development tools. The PC platform and Microsoft® Windows 3.x/95/98 environment were chosen to best make these features available to you, the end user.

MPLAB-ICE is available in two versions; MPLAB-ICE 1000 and MPLAB-ICE 2000. MPLAB-ICE 1000 is a basic, low-cost emulator system with simple trace capabilities. MPLAB-ICE 2000 is a full-featured emulator system with enhanced trace, trigger, and data monitoring features. Both systems use the same processor modules and will operate across the full operating speed range of the PIC MCU.

15.7 ICEPIC

ICEPIC is a low-cost in-circuit emulation solution for the Microchip Technology PIC16C5X, PIC16C6X, PIC16C7X, and PIC16CXXX families of 8-bit one-time-programmable (OTP) microcontrollers. The modular system can support different subsets of PIC16C5X or PIC16CXXX products through the use of interchangeable personality modules or daughter boards. The emulator is capable of emulating without target application circuitry being present.

15.8 MPLAB-ICD In-Circuit Debugger

Microchip's In-Circuit Debugger, MPLAB-ICD, is a powerful, low-cost run-time development tool. This tool is based on the flash PIC16F877 and can be used to develop for this and other PIC microcontrollers from the PIC16CXXX family. MPLAB-ICD utilizes the In-Circuit Debugging capability built into the PIC16F87X. This feature, along with Microchip's In-Circuit Serial Programming protocol, offers cost-effective in-circuit flash programming and debugging from the graphical user interface of the MPLAB Integrated Development Environment. This enables a designer to develop and debug source code by watching variables, single-stepping and setting break points. Running at full speed enables testing hardware in real-time. The MPLAB-ICD is also a programmer for the flash PIC16F87X family.

15.9 PRO MATE II Universal Programmer

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

The PRO MATE II 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 instructions and error messages, keys to enter commands and a modular detachable socket assembly to support various package types. In stand-alone mode the PRO MATE II can read, verify or program PIC devices. It can also set code-protect bits in this mode.

15.10 PICSTART Plus Entry Level Development System

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

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

15.11 PICDEM-1 Low-Cost PIC MCU Demonstration Board

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

15.12 PICDEM-2 Low-Cost PIC16CXX Demonstration Board

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

15.13 PICDEM-3 Low-Cost PIC16CXXX Demonstration Board

The PICDEM-3 is a simple demonstration board that supports the PIC16C923 and PIC16C924 in the PLCC package. It will also support future 44-pin PLCC microcontrollers with a LCD Module. All the necessary hardware and software is included to run the basic demonstration programs. The user can program the sample microcontrollers provided with the PICDEM-3 board, on a PRO MATE II programmer or PICSTART Plus with an adapter socket, and easily test firmware. The MPLAB-ICE emulator may also be used with the PICDEM-3 board to test firmware. Additional prototype area has been provided to the user for adding hardware and connecting it to the microcontroller socket(s). Some of the features include an RS-232 interface, push-button switches, a potentiometer for simulated analog input, a thermistor and separate headers for connection to an external LCD module and a keypad. Also provided on the PICDEM-3 board is an LCD panel, with 4 commons and 12 segments, that is capable of displaying time, temperature and day of the week. The PICDEM-3 provides an additional RS-232 interface and Windows 3.1 software for showing the demultiplexed LCD signals on a PC. A simple serial interface allows the user to construct a hardware demultiplexer for the LCD signals.

15.14 PICDEM-17

The PICDEM-17 is an evaluation board that demonstrates the capabilities of several Microchip microcontrollers, including PIC17C752, PIC17C756, PIC17C762, and PIC17C766. All necessary hardware is included to run basic demo programs, which are supplied on a 3.5-inch disk. A programmed sample is included, and the user may erase it and program it with the other sample programs using the PRO MATE II or PICSTART Plus device programmers and easily debug and test the sample code. In addition, PICDEM-17 supports down-loading of programs to and executing out of external FLASH memory on board. The PICDEM-17 is also usable with the MPLAB-ICE or PICMASTER emulator, and all of the sample programs can be run and modified using either emulator. Additionally, a generous prototype area is available for user hardware.

15.15 KEELOQ Evaluation and Programming Tools

KEELOQ evaluation and programming tools support Microchips HCS Secure Data Products. The HCS evaluation kit includes an LCD display to show changing codes, a decoder to decode transmissions, and a programming interface to program test transmitters.

TABLE 15-1: DEVELOPMENT TOOLS FROM MICROCHIP

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

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

** Contact Microchip Technology Inc. for availability date.

† Development tool is available on select devices.

NOTES:

16.0 ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings (†)

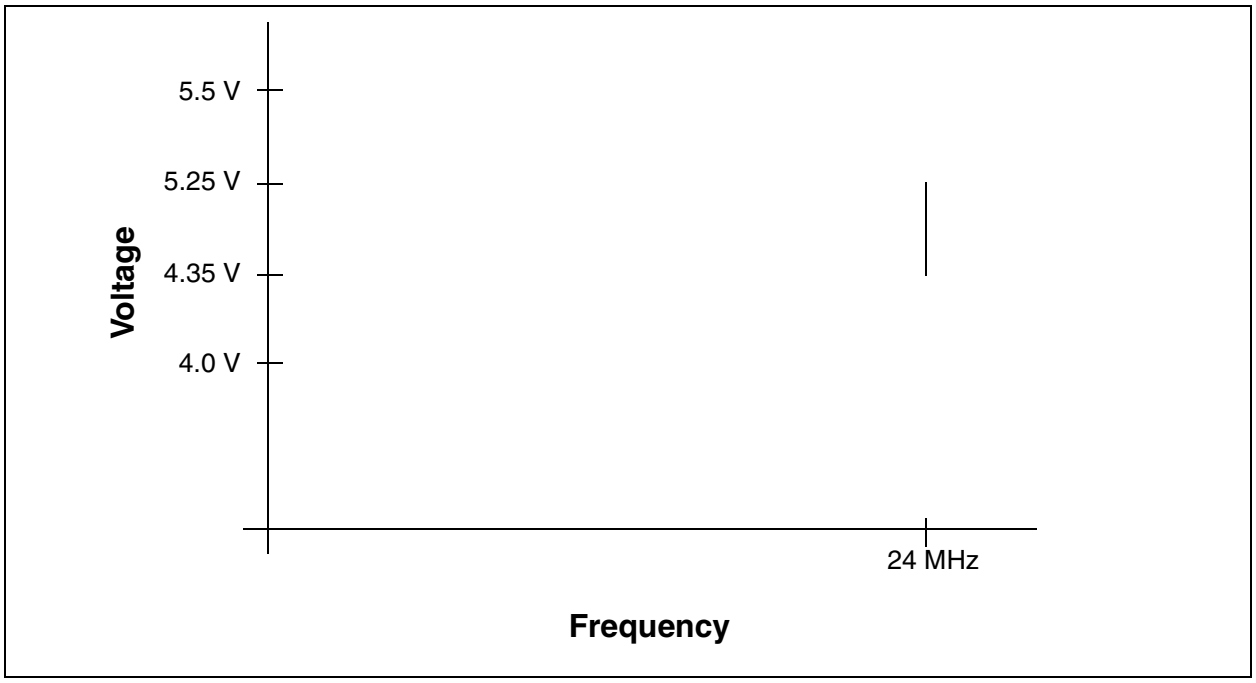
Ambient temperature under bias ..... -55°C to +125°C
Storage temperature ..... -65°C to +150°C
Voltage on any pin with respect to V_{SS} (except V_{DD}, $\overline{MCLR}$ and RA4) ..... -0.3V to (V_{DD} + 0.3V)
Voltage on V_{DD} with respect to V_{SS} ..... -0.3V to +7.5V
Voltage on $\overline{MCLR}$ with respect to V_{SS} ..... -0.3V to +13.25V
Voltage on RA4 with respect to Vss ..... -0.3V to +10.5V
Total power dissipation **(Note 1)** 1.0W
Maximum current out of V_{SS} pin ..... 300 mA
Maximum current into V_{DD} pin ..... 250 mA
Input clamp current, I_{IK} (V_I < 0 or V_I > V_{DD}) ..... ±20 mA
Output clamp current, I_{OK} (V_O < 0 or V_O > V_{DD}) ..... ±20 mA
Maximum output current sunk by any I/O pin ..... 25 mA
Maximum output current sourced by any I/O pin ..... 25 mA
Maximum current sunk by PORTA, PORTB, and PORTE **(Note 2)** (combined) ..... 200 mA
Maximum current sourced by PORTA, PORTB, and PORTE **(Note 2)** (combined) ..... 200 mA
Maximum current sunk by PORTC and PORTD **(Note 2)** (combined) ..... 200 mA
Maximum current sourced by PORTC and PORTD **(Note 2)** (combined) ..... 200 mA

Note 1: Power dissipation is calculated as follows: Pdis = V_{DD} x {I_{DD} - $\sum I_{OH}$} + $\sum$ {(V_{DD}-V_{OH}) x I_{OH}} + $\sum$(V_{OL} x I_{OL})

2: PORTD and PORTE not available on the PIC16C745.

† 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.

FIGURE 16-1: VALID OPERATING REGIONS, FREQUENCY ON FINT, -40°C ≤ TA ≤ +85°C



16.1 DC Characteristics: PIC16C745/765 (Industrial)

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial				
Param No.	**Sym**	**Characteristic**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
D001	VDD	**Supply Voltage**	4.35	—	5.25	V	See Figure 16-1
D002*	VDR	**RAM Data Retention Voltage (Note 1)**	—	1.5	—	V	
D003	VPOR	**VDD Start Voltage** to ensure internal Power-on Reset signal	—	VSS	—	V	See section on Power-on Reset for details
D004* D004A*	SVDD	**VDD Rise Rate** to ensure internal Power-on Reset signal	0.05 TBD	— —	— —	V/mS V/mS	PWRT enabled ($\overline{\text{PWRTE}}$ bit clear) PWRT disabled ($\overline{\text{PWRTE}}$ bit set) See section on Power-on Reset for details
D005	VBOR	**Brown-out Reset** voltage trip point	3.65	—	4.35	V	Brown-out Reset is always active
D010 D013	IDD	**Supply Current (Note 2, 4)**	— —	14 18	16 20	mA mA	FINT = 24 MHz, VDD = 4.35V FINT = 24 MHz, VDD = 5.25V
D020 D021 D021B	IPD	**Power-down Current (Note 3, 4)**	— —	90 100	120 140	µA µA	VDD = 4.35V w/ USB suspended VDD = 5.25V w/ USB suspended
		Module Differential Current (Note 5, 6)					
D022*	ΔIWDT	Watchdog Timer	—	6.0	20	µA	WDTE bit set, VDD = 4.35V
D022A*	ΔIUSB	Not suspend mode	—	40	180	µA	WDTE bit set, VDD = 4.35V
	ΔPLL	Phase Lock Loop	—	1.5	3.0	mA	WDTE bit set, VDD = 4.35V
1A	FOSC	HS oscillator operating freq.	24	—	24	MHz	All temperatures
		H4 oscillator operating freq.	6	—	6	MHz	All temperatures
		EC oscillator operating freq.	24	—	24	MHz	All temperatures
		E4 oscillator operating freq.	6	—	6	MHz	All temperatures

* 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.

Note 1: This is the limit to which VDD can be lowered 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,
$\overline{\text{MCLR}}$ = 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-impedance state and tied to VDD and VSS.

4: Timer1 oscillator (when enabled) adds approximately 20 µA to the specification. This value is from characterization and is for design guidance only. This is not tested.

5: The Δ current is the additional current consumed when this peripheral is enabled. This current should be added to the base IDD or IPD measurement.

6: Module differential currents measured at FINT = 24 MHz.

16.2 DC Characteristics: PIC16C745/765 (Industrial)

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial Operating voltage VDD range as described in DC spec Section 16.1 and Section 16.2				
Param No.	**Sym**	**Characteristic**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
		Input Low Voltage					
	VIL	I/O ports					
D030		with TTL buffer	VSS	—	0.8	V	
D030A							
D031		with Schmitt Trigger buffer	VSS	—	0.2 VDD	V	For entire VDD range
D032		MCLR, OSC1 (in EC, E4 mode)	VSS	—	0.2 VDD	V	
D033		OSC1 (in HS, H4 mode)	VSS	—	0.3 VDD	V	**Note 1**
		Input High Voltage					
	VIH	I/O ports		—			
D040		with TTL buffer	2.0	—	VDD	V	
D041		with Schmitt Trigger buffer	0.8 VDD	—	VDD	V	For entire VDD range
D042		MCLR	0.8 VDD	—	VDD	V	
D042A		OSC1 (HS, H4 mode)	0.7 VDD	—	VDD	V	**Note 1**
D043		OSC1 (in EC, E4 mode)	0.9 VDD	—	VDD	V	
		Input Leakage Current (Notes 2, 3)					
D060	IIL	I/O ports	—	—	±1	µA	VSS ≤ VPIN ≤ VDD, Pin at hi-impedance
D061		MCLR, RA4/T0CKI	—	—	±5	µA	VSS ≤ VPIN ≤ VDD
D063		OSC1	—	—	±5	µA	VSS ≤ VPIN ≤ VDD, HS osc mode
D070	IPURB	**PORTB weak pull-up current**	50	250	400	µA	VDD = 5V, VPIN = VSS
	AVIH	D+ In	2.4	—	—	V	VDD = 4.35V w/ USB suspended **(Note 4)**
		D- In	2.4	—	—	V	VDD = 4.35V w/ USB suspended
	AVOH	D+ Out	2.8	—	3.6	V	VDD = 4.35V w/ USB suspended **(Note 4)**
		D- Out	2.8	—	3.6	V	VDD = 4.35V w/ USB suspended
	AVIL	D+ In	—	—	.8	V	VDD = 4.35V w/ USB suspended **(Note 4)**
		D- In	—	—	.8	V	VDD = 4.35V w/ USB suspended
	AVOL	D+ Out	—	—	.3	V	VDD = 4.35V w/ USB suspended **(Note 4)**
		D- Out	—	—	.3	V	VDD = 4.35V w/ USB suspended
		Output Low Voltage					
D080	VOL	I/O ports	—	—	0.6	V	IOL = 8.5 mA, VDD = 4.35V, -40°C to +85°C
D083		OSC2/CLKOUT (EC, E4 osc mode)	—	—	0.6	V	IOL = 1.6 mA, VDD = 4.35V, -40°C to +85°C
		Output High Voltage					
D090	VOH	I/O ports **(Note 3)**	VDD-0.7	—	—	V	IOH = -3.0 mA, VDD = 4.35V, -40°C to +85°C
D092		OSC2/CLKOUT (EC osc mode)	VDD-0.7	—	—	V	IOH = -1.3 mA, VDD = 4.35V, -40°C to +85°C
D150*	VOD	**Open-Drain High Voltage**	—	—	10.5	V	RA4 pin
		Capacitive Loading Specs on Output Pins					
D100	COSC2	OSC2 pin	—	—	15	pF	In HS mode when external clock is used to drive OSC1.
D101	CIO	All I/O pins and OSC2 (in EC mode)	—	—	50	pF	
	CVUSB	VUSB regulation capacitor	—	200	—	nF	± 20% (See Section 10.7.1)

*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.

Note 1: In EC oscillator mode, the OSC1/CLKIN pin is a Schmitt Trigger input.

2: The leakage current on the MCLR/VPP 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 current sourced by the pin.

4: Parameters are per USB Specification 1.1. No Microchip specific parameter numbers exist (per the PIC Mid-Range Reference Manual, DS33023.

16.3 AC (Timing) Characteristics

16.3.1 TIMING PARAMETER SYMBOLOGY

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

1. TppS2ppS
2. TppS

T			
F	Frequency	T	Time

Lowercase letters (pp) and their meanings:

pp			
cc	CCP1	osc	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	$\overline{WR}$

Uppercase letters and their meanings:

S			
F	Fall	P	Period
H	High	R	Rise
I	Invalid (Hi-impedance)	V	Valid
L	Low	Z	Hi-impedance

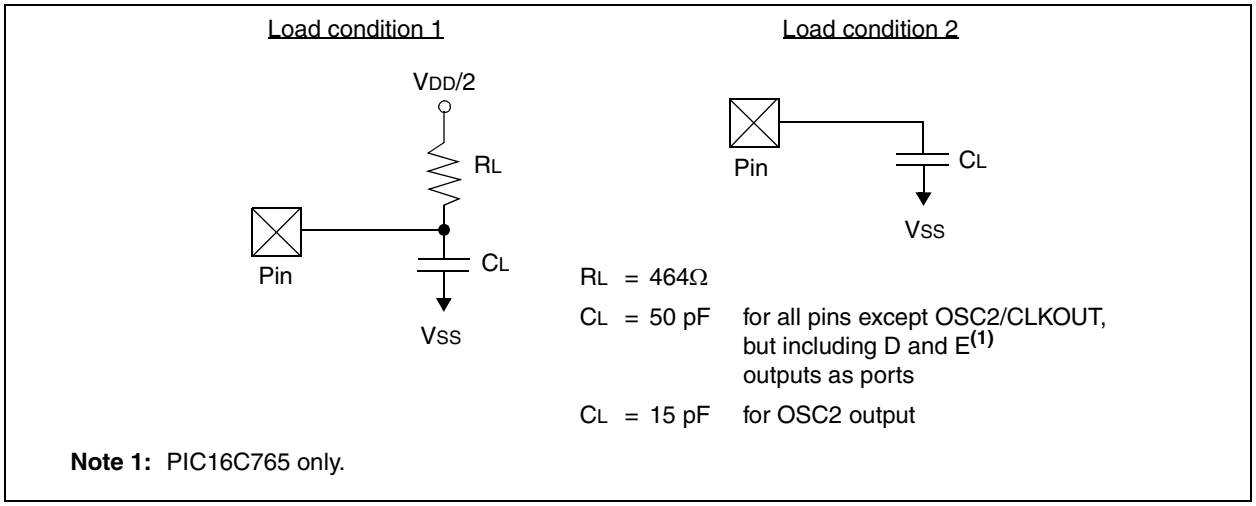
16.3.2 TIMING CONDITIONS

The temperature and voltages specified in Table 16-1 apply to all timing specifications unless otherwise noted. Figure 16-2 specifies the load conditions for the timing specifications.

TABLE 16-1: TEMPERATURE AND VOLTAGE SPECIFICATIONS - AC

AC CHARACTERISTICS	**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial Operating voltage VDD range as described in DC spec Section 16.1 and Section 16.2.

FIGURE 16-2: LOAD CONDITIONS FOR DEVICE TIMING SPECIFICATIONS



Note 1: PIC16C765 only.

16.3.3 TIMING DIAGRAMS AND SPECIFICATIONS

FIGURE 16-3: EXTERNAL CLOCK TIMING

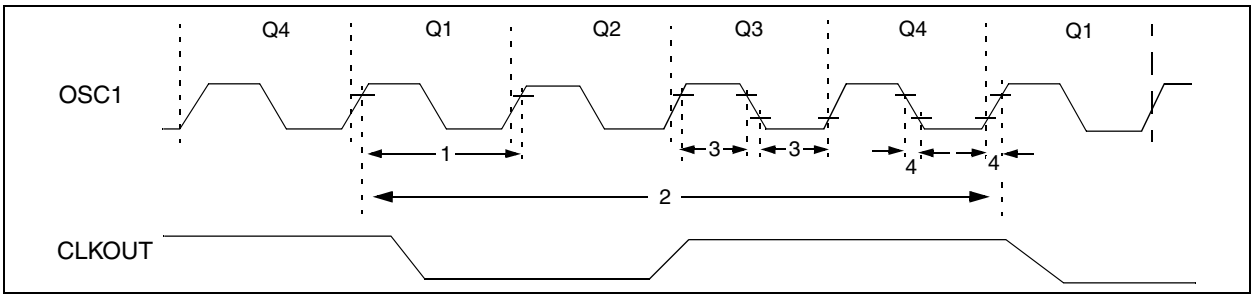
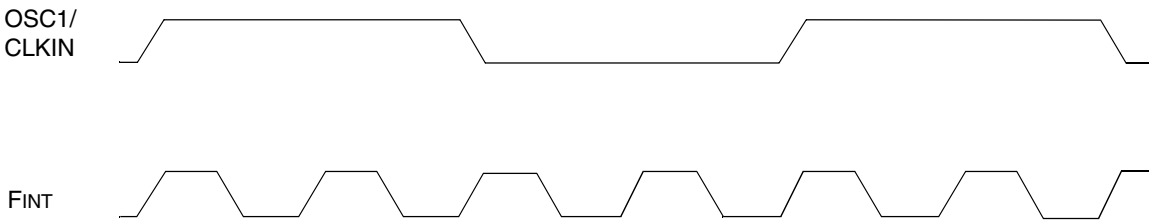


FIGURE 16-4: CLOCK MULTIPLIER (PLL) PHASE RELATIONSHIP



Note 1: FINT represents the internal clock signal. FINT equals FOSC or CLKIN if the PLL is disabled. FINT equals 4x FOSC or 4x CLKIN if the PLL is enabled. TCY is always 4/FINT. FINT is OSC1 pin in EC mode, PLL disabled.

2: FINT = OSC1 in EC mode with PLL disabled.

TABLE 16-2: EXTERNAL CLOCK TIMING REQUIREMENTS

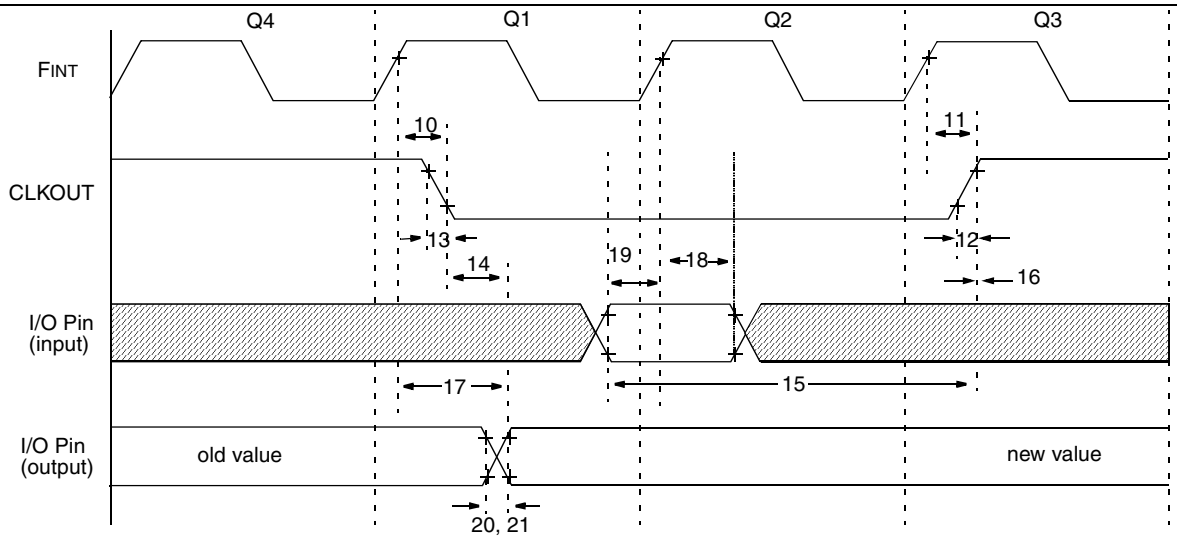
Param No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
1A	FOSC	External CLKIN Frequency (Note 1)	24	—	24	MHz	EC osc mode
			6	—	6	MHz	E4 osc mode
		Oscillator Frequency (Note 1)	24	—	24	MHz	HS osc mode
			6	—	6	MHz	H4 osc mode
1	TOSC	External CLKIN Period (Note 1)	41	—	41	ns	EC osc modes
			167	—	167	ns	E4 osc mode
		Oscillator Period (Note 1)	41	—	41	ns	HS osc modes
			167	—	167	ns	H4 osc mode
2	TCY	Instruction Cycle Time (Note 1)	167	—	DC	ns	TCY = 4/FINT
3*	TOSL, TOSH	External Clock in (OSC1) High or Low Time	10	—	—	ns	EC oscillator
4*	TOSR, TOSF	External Clock in (OSC1) Rise or Fall Time	—	—	15	ns	EC oscillator

* 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.

Note 1: Instruction cycle period (TCY) equals four times the input oscillator time-base period when the PLL is enabled, or the input oscillator time-base period divided by 4 when the PLL is disabled. 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/CLKIN pin. When an external clock input is used, the "Max." cycle time limit is "DC" (no clock) for all devices.

FIGURE 16-5: CLKOUT AND I/O TIMING



Note: Refer to Figure 16-2 for load conditions.

TABLE 16-3: CLKOUT AND I/O TIMING REQUIREMENTS

Param No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
10*	TOSH2CKL	OSC1↑ to CLKOUT↓	—	75	200	ns	Note 1
11*	TOSH2CKH	OSC1↑ to CLKOUT↑	—	75	200	ns	Note 1
12*	TCKR	CLKOUT rise time	—	35	100	ns	Note 1
13*	TCKF	CLKOUT fall time	—	35	100	ns	Note 1
14*	TCKL2IOV	CLKOUT ↓ to Port out valid	—	—	0.5 TCY + 20	ns	Note 1
15*	TIOV2CKH	Port in valid before CLKOUT ↑	TOSC + 200	—	—	ns	Note 1
16*	TCKH2IOI	Port in hold after CLKOUT ↑	0	—	—	ns	Note 1
17*	TOSH2IOV	OSC1↑ (Q1 cycle) to Port out valid	—	50	150	ns	
18*	TOSH2IOI	OSC1↑ (Q2 cycle) to Port input invalid (I/O in hold time)	100	—	—	ns	
19*	TIOV2OSH	Port input valid to OSC1↑ (I/O in setup time)	0	—	—	ns	
20*	TIOR	Port output rise time	—	10	40	ns	
21*	TIOF	Port output fall time	—	10	40	ns	
22††*	TINP	INT pin high or low time	TCY	—	—	ns	
23††*	TRBP	RB<7:4> change INT high or low time	TCY	—	—	ns	

* These parameters are characterized but not tested.

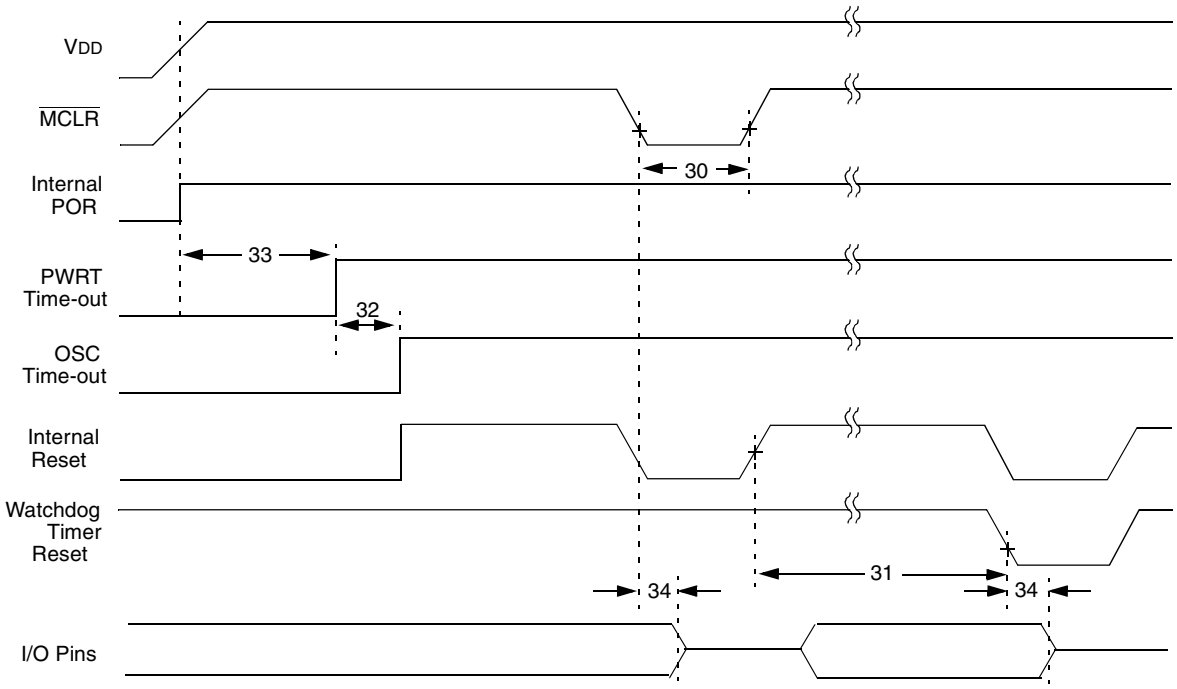
†Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

††These parameters are asynchronous events not related to any internal clock edge.

Note 1: Measurements are taken in EC Mode where CLKOUT output is 4 x TOSC.

2: FINT = OSC1 when PLL is disabled.

FIGURE 16-6: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER TIMING



Note: Refer to Figure 16-2 for load conditions.

FIGURE 16-7: BROWN-OUT RESET TIMING

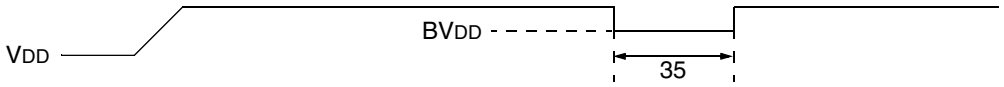


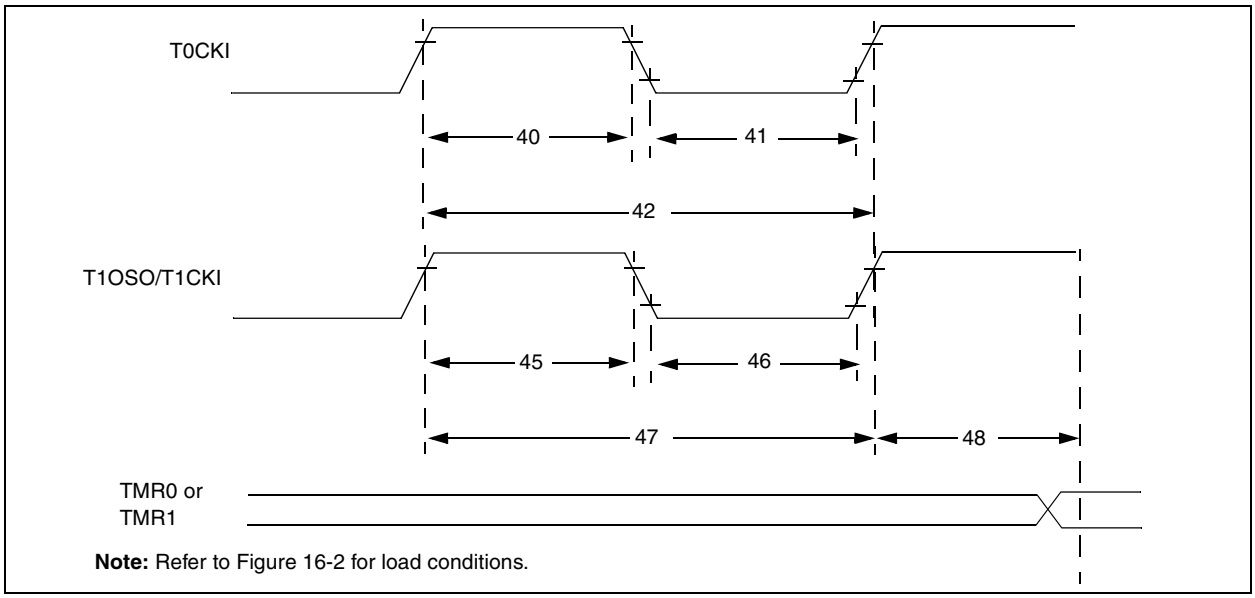
TABLE 16-4: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER, POWER-UP TIMER, AND BROWN-OUT RESET REQUIREMENTS

Param No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
30	TMCL	$\overline{MCLR}$ Pulse Width (low)	2	—	—	µs	VDD = 5V, -40°C to +85°C
31*	TWDT	Watchdog Timer Time-out Period (No Prescaler)	7	18	33	ms	VDD = 5V, -40°C to +85°C
32	TOST	Oscillation Start-up Timer Period	—	1024 TOSC	—	—	TOSC = OSC1 period
33*	TPWRT	Power-up Timer Period	28	72	132	ms	VDD = 5V, -40°C to +85°C
34	TIOZ	I/O Hi-impedance from $\overline{MCLR}$ Low or WDT Reset	—	—	2.1	µs	
35	TBOR	Brown-out Reset Pulse Width	100	—	—	µs	VDD ≤ BVDD (D005)
36	TPLLRT	PLL Settling Time Period	—	1.4	—	ms	TOSC = OSC1 period

* 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.

FIGURE 16-8: TIMER0 AND TIMER1 EXTERNAL CLOCK TIMINGS



Note: Refer to Figure 16-2 for load conditions.

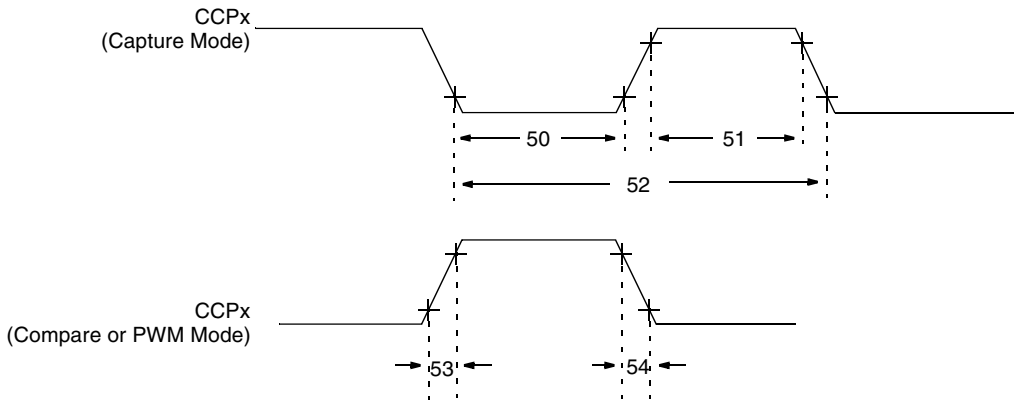
TABLE 16-5: TIMER0 AND TIMER1 EXTERNAL CLOCK REQUIREMENTS

Param No.	Sym	Characteristic		Min	Typ†	Max	Units	Conditions
40*	TT0H	T0CKI High Pulse Width	No Prescaler	0.5 TCY + 20	—	—	ns	Must also meet parameter 42
			With Prescaler	10	—	—	ns	
41*	TT0L	T0CKI Low Pulse Width	No Prescaler	0.5 TCY + 20	—	—	ns	Must also meet parameter 42
			With Prescaler	10	—	—	ns	
42*	TT0P	T0CKI Period	No Prescaler	TCY + 40	—	—	ns	N = prescale value (2, 4,..., 256)
			With Prescaler	Greater of: 20 or TCY + 40 / N	—	—	ns	
45*	TT1H	T1CKI High Time	Synchronous, Prescaler = 1	0.5 TCY + 20	—	—	ns	Must also meet parameter 47
			Synchronous, Prescaler = 2,4,8	15	—	—	ns	
			Asynchronous	30	—	—	ns	
46*	TT1L	T1CKI Low Time	Synchronous, Prescaler = 1	0.5 TCY + 20	—	—	ns	Must also meet parameter 47
			Synchronous, Prescaler = 2,4,8	15	—	—	ns	
			Asynchronous	30	—	—	ns	
47*	TT1P	T1CKI input period	Synchronous	Greater of: 30 or TCY + 40 / N	—	—	ns	N = prescale value (1, 2, 4, 8)
			Asynchronous	60	—	—	ns	
	FT1	Timer1 oscillator input frequency range (oscillator enabled by setting bit T1OSCEN)		DC	—	200	kHz	
48	TCKEZTMR1	Delay from external clock edge to timer increment		2Tosc	—	7Tosc	—	

* 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.

FIGURE 16-9: CAPTURE/COMPARE/PWM TIMINGS (CCP1 AND CCP2)



Note: Refer to Figure 16-2 for load conditions.

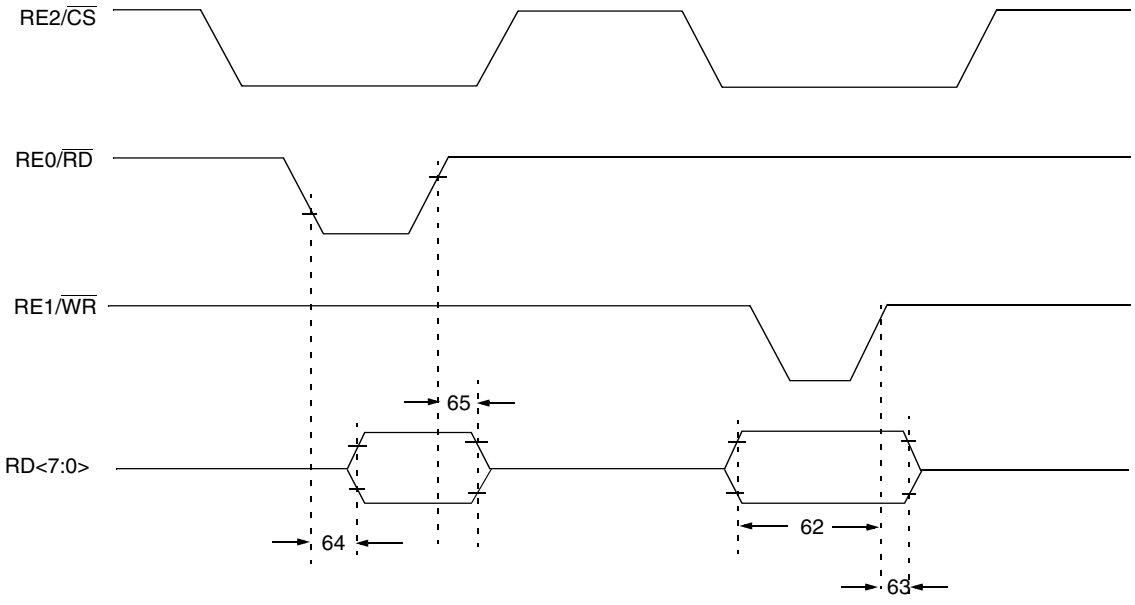
TABLE 16-6: CAPTURE/COMPARE/PWM REQUIREMENTS (CCP1 AND CCP2)

Param No.	Sym	Characteristic		Min	Typ†	Max	Units	Conditions
50*	TccL	CCP1 and CCP2 input low time	No Prescaler	0.5 TCY + 20	—	—	ns	
			With Prescaler	10	—	—	ns	
51*	TccH	CCP1 and CCP2 input high time	No Prescaler	0.5 TCY + 20	—	—	ns	
			With Prescaler	10	—	—	ns	
52*	TccP	CCP1 and CCP2 input period		$\frac{3\ T_{CY} + 40}{N}$	—	—	ns	N = prescale value (1,4, or 16)
53*	TccR	CCP1 and CCP2 output rise time		—	10	25	ns	
54*	TccF	CCP1 and CCP2 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.

FIGURE 16-10: PARALLEL SLAVE PORT TIMING (PIC16C765)



Note: Refer to Figure 16-2 for load conditions.

TABLE 16-7: PARALLEL SLAVE PORT REQUIREMENTS

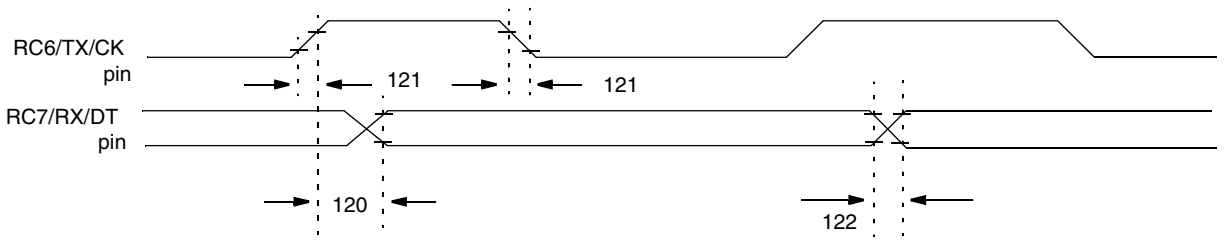
Param No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
62*	TDTV2WRH	Data in valid before $\overline{WR}\uparrow$ or $\overline{CS}\uparrow$ (setup time)	20	—	—	ns	
63*	TWRH2DTI	$\overline{WR}\uparrow$ or $\overline{CS}\uparrow$ to data–in invalid (hold time)	20	—	—	ns	
64	TRDL2DTV	$\overline{RD}\downarrow$ and $\overline{CS}\downarrow$ to data–out valid	—	—	80	ns	
65*	TRDH2DTI	$\overline{RD}\uparrow$ or $\overline{CS}\uparrow$ to data–out invalid	10	—	30	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.

Note: PIC16C765 only.

FIGURE 16-11: USART SYNCHRONOUS TRANSMISSION (MASTER/SLAVE) TIMING



Note: Refer to Figure 16-2 for load conditions.

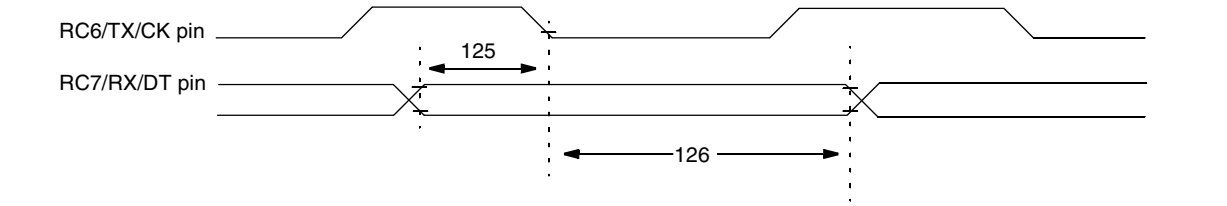
TABLE 16-8: USART SYNCHRONOUS TRANSMISSION REQUIREMENTS

Param No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
120*	TCKH2DTV	SYNC XMIT (MASTER & SLAVE) Clock high to data out valid	—	—	80	ns	
121*	TCKRF	Clock out rise time and fall time (Master mode)	—	—	45	ns	
122*	TDTRF	Data out rise time and fall time	—	—	45	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.

FIGURE 16-12: USART SYNCHRONOUS RECEIVE (MASTER/SLAVE) TIMING



Note: Refer to Figure 16-2 for load conditions.

TABLE 16-9: USART SYNCHRONOUS RECEIVE REQUIREMENTS

Param No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
125*	TDTV2CKL	SYNC RCV (MASTER & SLAVE) Data setup before CK ↓ (DT setup time)	15	—	—	ns	
126*	TCKL2DTL	Data hold after CK ↓ (DT hold time)	15	—	—	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.

TABLE 16-10: A/D CONVERTER CHARACTERISTICS: PIC16C745/765 (INDUSTRIAL)

Param No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
A01	NR	Resolution	—	—	8 bits	bit	
A02	EABS	Total Absolute error	—	—	< ± 1	LSb	VREF = VDD = 5.12V, VSS ≤ VAIN ≤ VREF
A03	EIL	Integral linearity error	—	—	< ± 1	LSb	VREF = VDD = 5.12V, VSS ≤ VAIN ≤ VREF
A04	EDL	Differential linearity error	—	—	< ± 1	LSb	VREF = VDD = 5.12V, VSS ≤ VAIN ≤ VREF
A05	EFS	Full scale error	—	—	< ± 1	LSb	VREF = VDD = 5.12V, VSS ≤ VAIN ≤ VREF
A06	EOFF	Offset error	—	—	< ± 1	LSb	VREF = VDD = 5.12V, VSS ≤ VAIN ≤ VREF
A10	—	Monotonicity **(Note 3)**	—	warranteed	—	—	VSS ≤ VAIN ≤ VREF
A20	VREF	Reference voltage	2.5V	—	VDD + 0.3	V	
A25	VAIN	Analog input voltage	VSS - 0.3	—	VREF + 0.3	V	
A30	ZAIN	Recommended impedance of analog voltage source	—	—	10.0	kΩ	
A40	IAD	A/D conversion current (VDD)	—	180	—	μA	Average current consumption when A/D is on **(Note 1)**
A50	IREF	VREF input current **(Note 2)**	10 —	— —	1000 10	μA μA	During VAIN acquisition. Based on differential of VHOLD to VAIN to charge CHOLD, see Section 12.1. During A/D Conversion cycle

* These parameters are characterized but not tested.

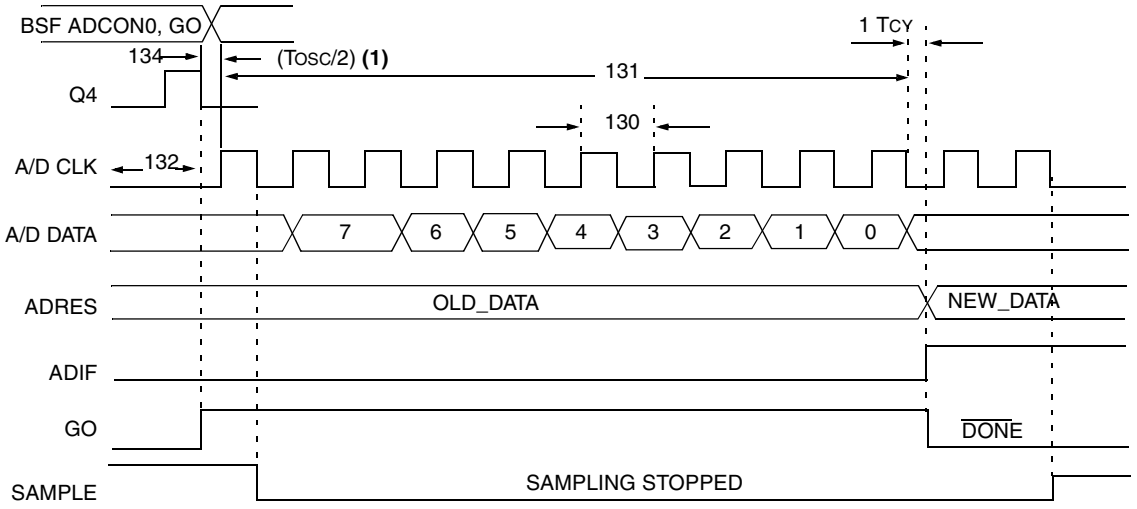
† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: When A/D is off, it will not consume any current other than minor leakage current. The power-down current spec includes any such leakage from the A/D module.

2: VREF current is from the RA3 pin or the VDD pin, whichever is selected as a reference input.

3: The A/D conversion result never decreases with an increase in the input voltage and has no missing codes.

FIGURE 16-13: A/D CONVERSION TIMING



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

TABLE 16-11: A/D CONVERSION REQUIREMENTS

Param No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
130	TAD	A/D clock period	1.6	—	—	µs	TOSC based, VREF ≥ 3.0V
			2.0	—	—	µs	TOSC based, 2.5V ≤ VREF ≤ 5.5V
			2.0	4.0	6.0	µs	A/D RC Mode
131	TCNV	Conversion time (not including S/H time) **(Note 1)**	11	—	11	TAD	
132	TACQ	Acquisition time	5*	—	—	µs	The minimum time is the amplifier settling time. This may be used if the "new" input voltage has not changed by more than 1 LSb (i.e., 20.0 mV @ 5.12V) from the last sampled voltage (as stated on CHOLD).
134	TGO	Q4 to A/D clock start	—	TOSC/2	—	—	If the A/D clock source is selected as RC, a time of TCY is added before the A/D clock starts. This allows the SLEEP instruction to be executed.
135	TSWC	Switching from convert → sample time	1.5	—	—	TAD	

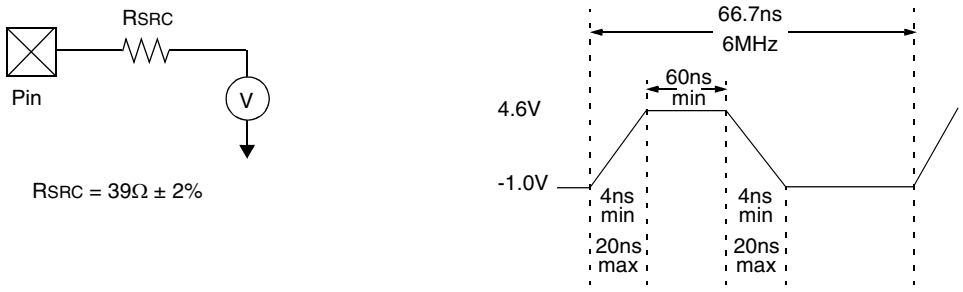
* These parameters are characterized but not tested.

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

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

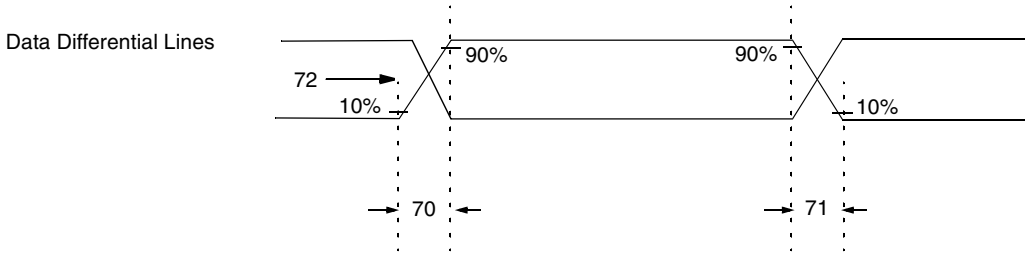
2: See Section 12.1 for minimum conditions.

FIGURE 16-14: MAXIMUM INPUT WAVEFORM TIMING SPECIFICATIONS



Note 1: The D+/D- signals can withstand a continuous short to VBUS, GND, cable shield or any other signal.

FIGURE 16-15: USB LOW SPEED SIGNALING



Note 1: Parameters are per USB Specification 1.1. No Microchip specific parameter numbers exist (per the PIC Mid-Range Reference Manual, DS33023.

TABLE 16-12: USB AC AND DC SPECIFICATIONS

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
	TLR	Transition Rise Time	75		300	ns	**(Note 1)**
	TLF	Transition Fall Time	75		300	ns	**(Note 1)**
	VCRS	Crossover Voltage	1.3		2.0	V	**(Note 1)**
	TLRFM	Rise and Fall Time Matching	80		125	%	**(Note 1)**
	VIL	Voltage Input Low			0.8	V	**(Note 1)**
	VIH	Voltage Input High	2.0			V	**(Note 1)**
	VIHZ	Voltage Input High Floating	2.7		3.6	V	**(Note 1)**
		Differential Input Sensitivity	0.2			V	(D+)-(D-) **(Note 1)**
		Differential Common Mode Range	0.8		2.5	V	**(Note 1)**
	VOL	Voltage Output Low	0.0		0.3	V	**(Note 1)**
	VOH	Voltage Output High	2.8		3.6	V	**(Note 1)**
	VUSB	USB Voltage Output	2.7		3.6	V	IUSB = 0 to 230μA in suspend mode 0 to 2.6mA in non-suspend mode **(Note 1)**

Note 1: Parameters are per USB Specification 1.1. No Microchip specific parameter numbers exist (per the PIC Mid-Range Reference Manual, DS33023.

NOTES:

17.0 DC AND AC CHARACTERISTICS GRAPHS AND TABLES

The graphs and tables provided in this section are for design guidance and are not tested. In some graphs or tables, the data presented are outside specified operating range. This is for information only and devices will operate properly only within the specified range.

The data presented in this section is a statistical summary of data collected on units from different lots over a period of time. “Typical” represents the mean of the distribution.

FIGURE 17-1: TYPICAL IDD vs. VDD (FINT = 24MHz)

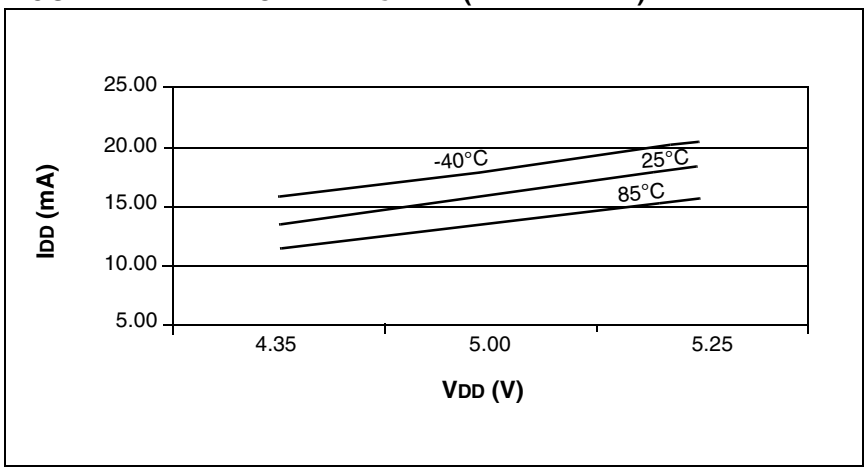


FIGURE 17-2: TYPICAL IPD vs. VDD (USB SUSPENDED, WDT DISABLED)

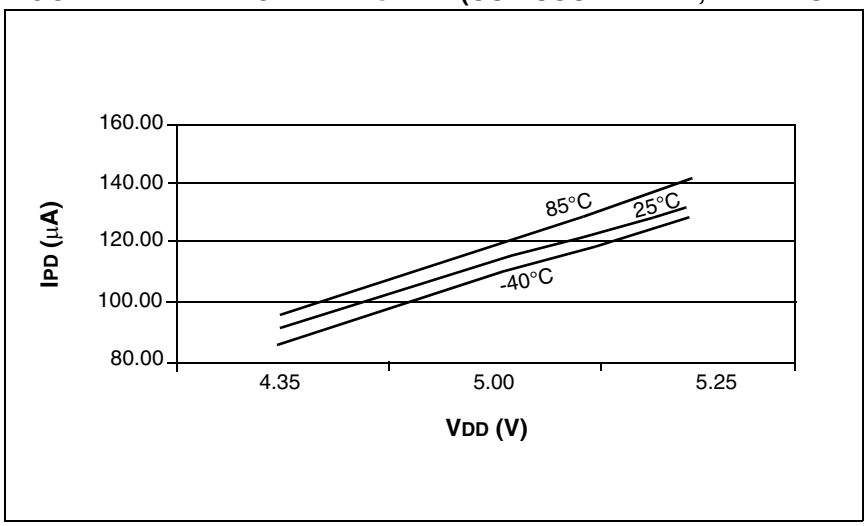
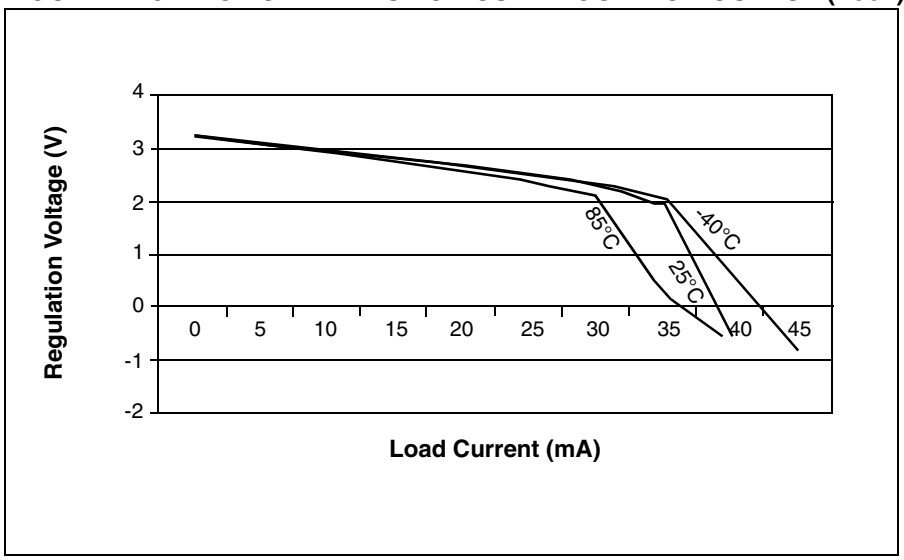


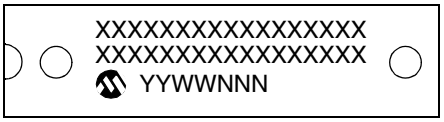
FIGURE 17-3: DC LOAD LINES FOR USB REGULATOR OUTPUT (VUSB)



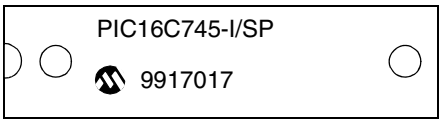
18.0 PACKAGING INFORMATION

18.1 Package Marking Information

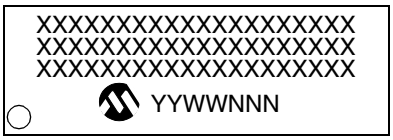
28-Lead PDIP (Skinny DIP)



Example

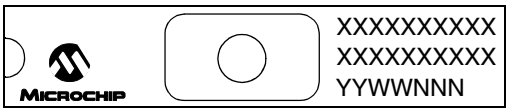


28-Lead SOIC

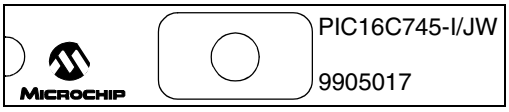


Example

28-Lead Side Braze Windowed (JW)



Example

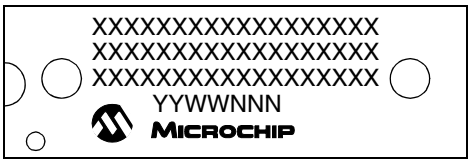


Legend:	XX...X	Customer-specific information
	Y	Year code (last digit of calendar year)
	YY	Year code (last 2 digits of calendar year)
	WW	Week code (week of January 1 is week ‘01’)
	NNN	Alphanumeric traceability code
	(e3)	Pb-free JEDEC designator for Matte Tin (Sn)
	*	This package is Pb-free. The Pb-free JEDEC designator ((e3)) can be found on the outer packaging for this package.

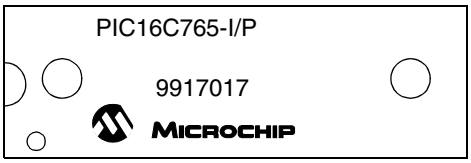
Note: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.

Package Marking Information (Cont'd)

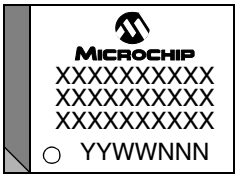
40-Lead PDIP



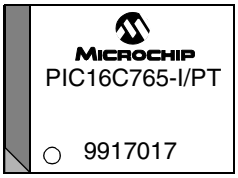
Example



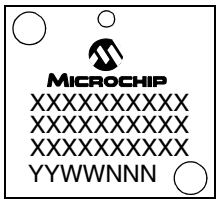
44-Lead TQFP



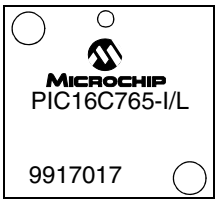
Example



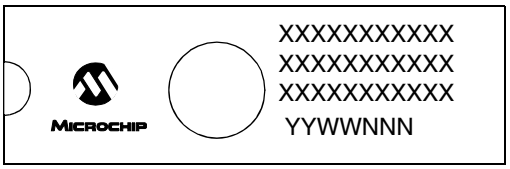
44-Lead PLCC



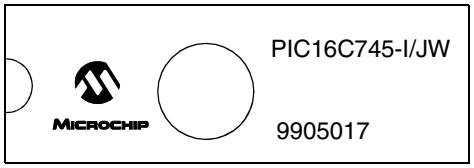
Example



40-Lead CERDIP Windowed

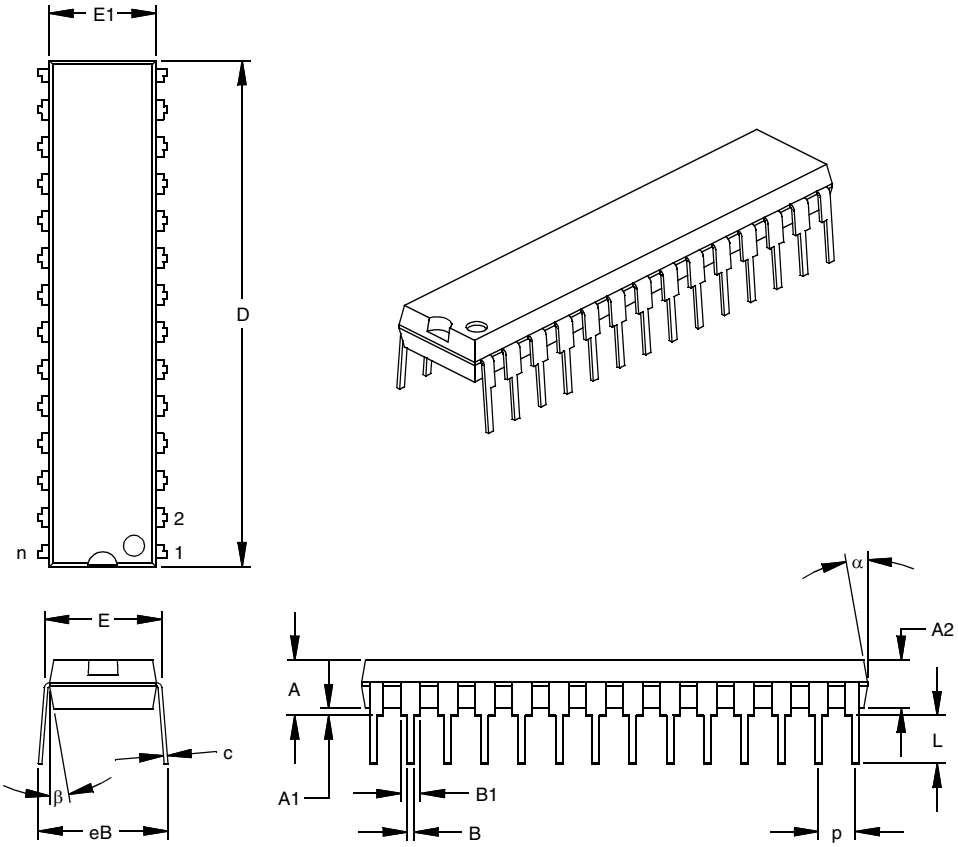


Example



28-Lead Skinny Plastic Dual In-line (SP) – 300 mil (PDIP)

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging

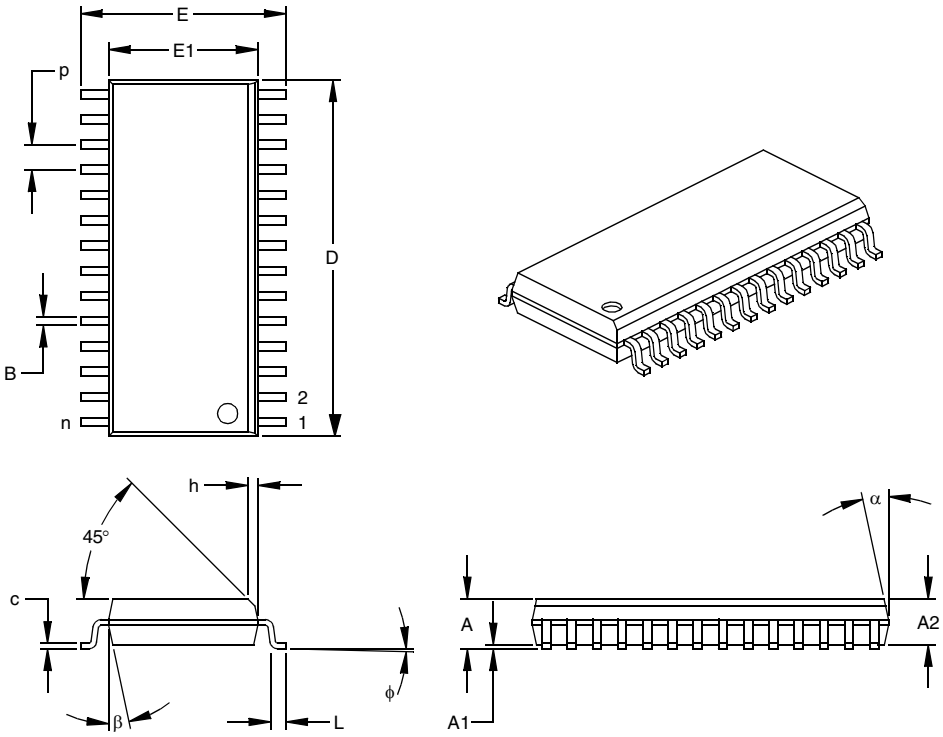


Units		INCHES*			MILLIMETERS		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		28			28	
Pitch	p		.100			2.54	
Top to Seating Plane	A	.140	.150	.160	3.56	3.81	4.06
Molded Package Thickness	A2	.125	.130	.135	3.18	3.30	3.43
Base to Seating Plane	A1	.015			0.38		
Shoulder to Shoulder Width	E	.300	.310	.325	7.62	7.87	8.26
Molded Package Width	E1	.275	.285	.295	6.99	7.24	7.49
Overall Length	D	1.345	1.365	1.385	34.16	34.67	35.18
Tip to Seating Plane	L	.125	.130	.135	3.18	3.30	3.43
Lead Thickness	c	.008	.012	.015	0.20	0.29	0.38
Upper Lead Width	B1	.040	.053	.065	1.02	1.33	1.65
Lower Lead Width	B	.016	.019	.022	0.41	0.48	0.56
Overall Row Spacing §	eB	.320	.350	.430	8.13	8.89	10.92
Mold Draft Angle Top	α	5	10	15	5	10	15
Mold Draft Angle Bottom	β	5	10	15	5	10	15

* Controlling Parameter
§ Significant Characteristic
Notes:
Dimension D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" (0.254mm) per side.
JEDEC Equivalent: MO-095
Drawing No. C04-070

28-Lead Plastic Small Outline (SO) – Wide, 300 mil (SOIC)

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



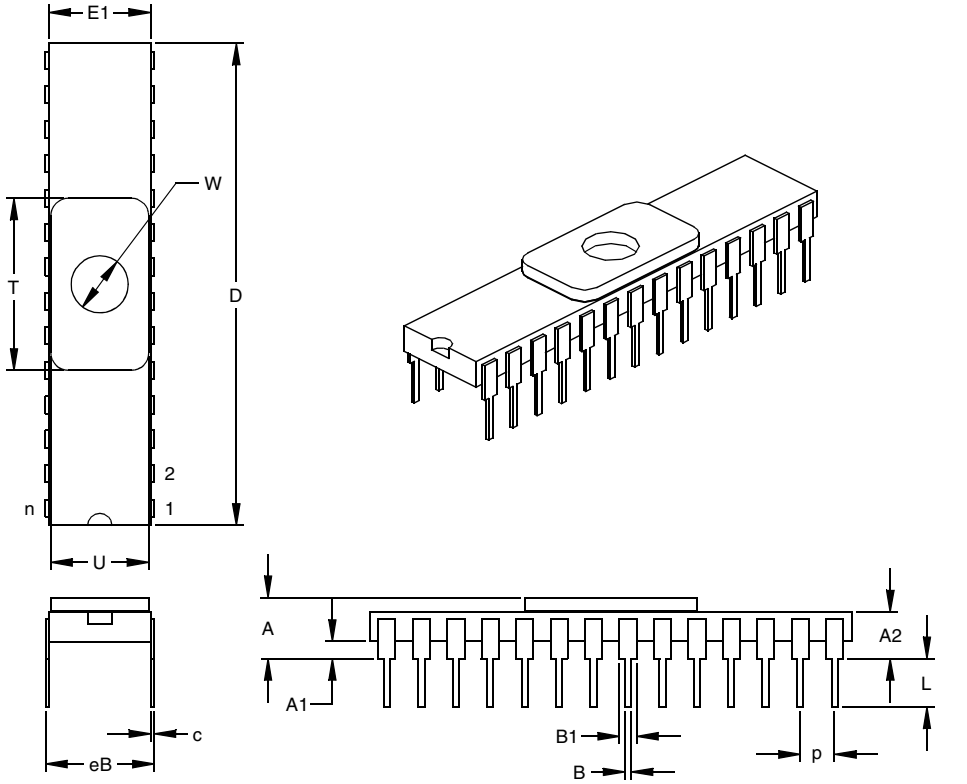
Units		INCHES*			MILLIMETERS		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		28			28	
Pitch	p		.050			1.27	
Overall Height	A	.093	.099	.104	2.36	2.50	2.64
Molded Package Thickness	A2	.088	.091	.094	2.24	2.31	2.39
Standoff §	A1	.004	.008	.012	0.10	0.20	0.30
Overall Width	E	.394	.407	.420	10.01	10.34	10.67
Molded Package Width	E1	.288	.295	.299	7.32	7.49	7.59
Overall Length	D	.695	.704	.712	17.65	17.87	18.08
Chamfer Distance	h	.010	.020	.029	0.25	0.50	0.74
Foot Length	L	.016	.033	.050	0.41	0.84	1.27
Foot Angle Top	ϕ	0	4	8	0	4	8
Lead Thickness	c	.009	.011	.013	0.23	0.28	0.33
Lead Width	B	.014	.017	.020	0.36	0.42	0.51
Mold Draft Angle Top	α	0	12	15	0	12	15
Mold Draft Angle Bottom	β	0	12	15	0	12	15

* Controlling Parameter
§ Significant Characteristic

Notes:
Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010” (0.254mm) per side.
JEDEC Equivalent: MS-013
Drawing No. C04-052

28-Lead Ceramic Side Brazed Dual In-line with Window (JW) – 300 mil

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging

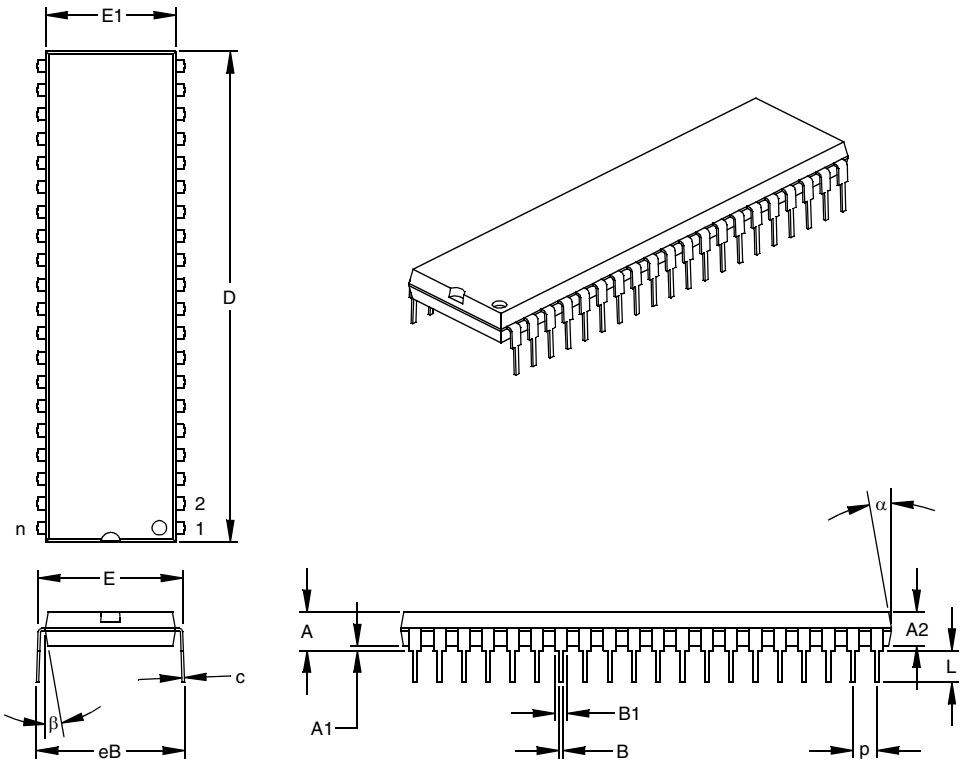


Units		INCHES*			MILLIMETERS		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		28			28	
Pitch	p		.100			2.54	
Top to Seating Plane	A	.155	.177	.198	3.94	4.48	5.03
Top of Body to Seating Plane	A2	.115	.135	.155	2.92	3.43	3.94
Standoff	A1	.040	.050	.060	1.02	1.27	1.52
Package Width	E1	.280	.290	.300	7.11	7.37	7.62
Overall Length	D	1.386	1.400	1.414	35.20	35.56	35.92
Tip to Seating Plane	L	.130	.140	.150	3.30	3.56	3.81
Lead Thickness	c	.008	.010	.012	0.20	0.25	0.30
Upper Lead Width	B1	.048	.050	.052	1.22	1.27	1.32
Lower Lead Width	B	.016	.018	.020	0.41	0.46	0.51
Overall Row Spacing §	eB	.296	.310	.324	7.52	7.87	8.23
Window Diameter	W	.161	.166	.171	4.09	4.22	4.34
Lid Length	T	.490	.500	.510	12.45	12.70	12.95
Lid Width	U	.275	.285	.295	6.99	7.24	7.49

* Controlling Parameter
§ Significant Characteristic
JEDEC Equivalent: MS-015
Drawing No. C04-084

40-Lead Plastic Dual In-line (P) – 600 mil (PDIP)

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging

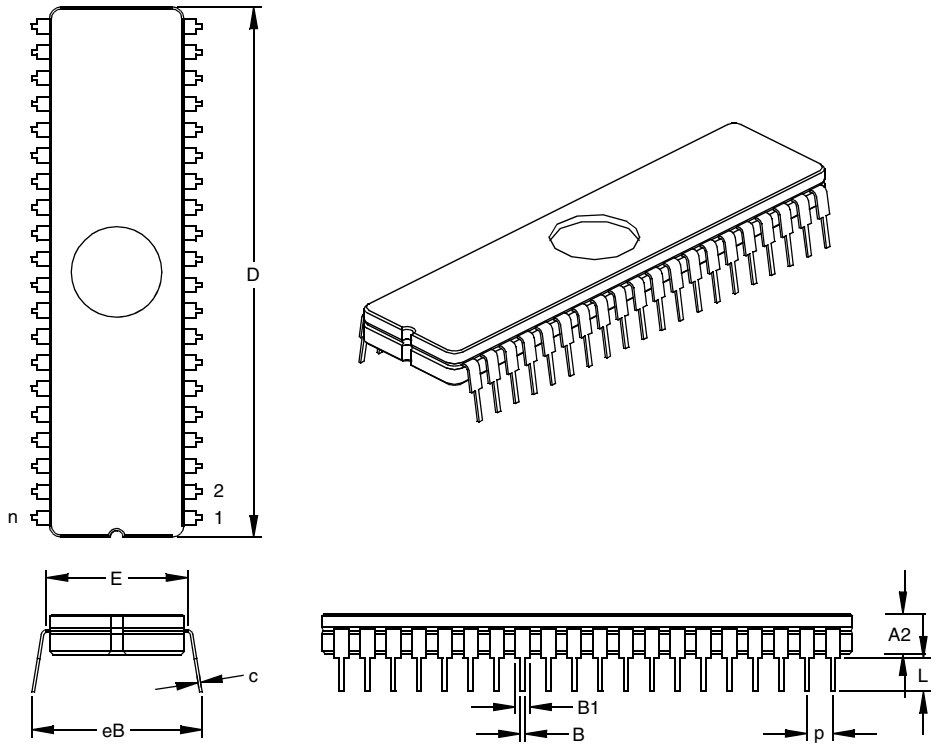


Units		INCHES*			MILLIMETERS		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		40			40	
Pitch	p		.100			2.54	
Top to Seating Plane	A	.160	.175	.190	4.06	4.45	4.83
Molded Package Thickness	A2	.140	.150	.160	3.56	3.81	4.06
Base to Seating Plane	A1	.015			0.38		
Shoulder to Shoulder Width	E	.595	.600	.625	15.11	15.24	15.88
Molded Package Width	E1	.530	.545	.560	13.46	13.84	14.22
Overall Length	D	2.045	2.058	2.065	51.94	52.26	52.45
Tip to Seating Plane	L	.120	.130	.135	3.05	3.30	3.43
Lead Thickness	c	.008	.012	.015	0.20	0.29	0.38
Upper Lead Width	B1	.030	.050	.070	0.76	1.27	1.78
Lower Lead Width	B	.014	.018	.022	0.36	0.46	0.56
Overall Row Spacing §	eB	.620	.650	.680	15.75	16.51	17.27
Mold Draft Angle Top	α	5	10	15	5	10	15
Mold Draft Angle Bottom	β	5	10	15	5	10	15

* Controlling Parameter
§ Significant Characteristic
Notes:
Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010” (0.254mm) per side.
JEDEC Equivalent: MO-011
Drawing No. C04-016

40-Lead Ceramic Dual In-line with Window (JW) – 600 mil (CERDIP)

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging

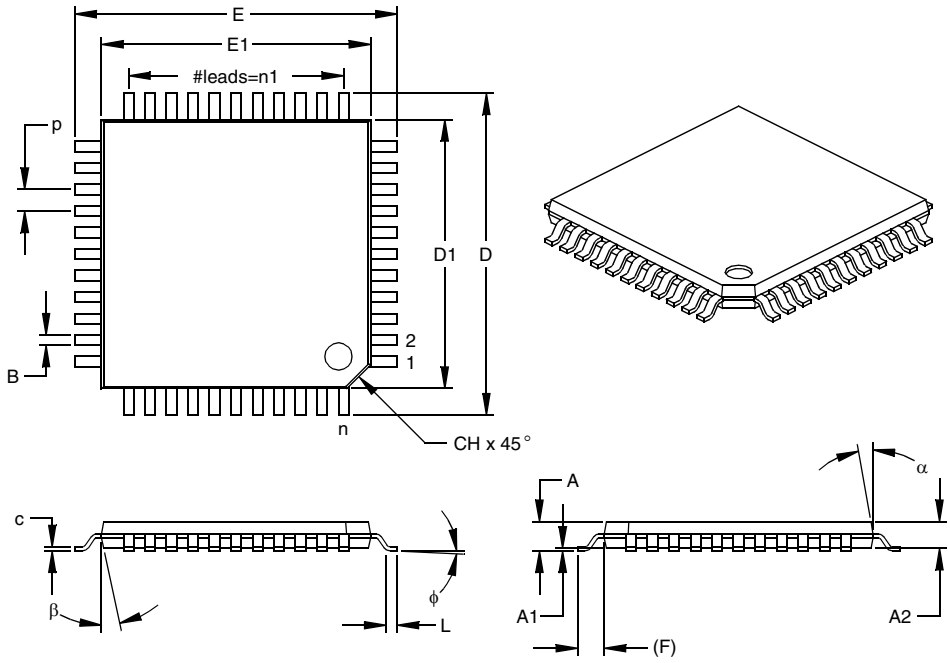


Units		INCHES*			MILLIMETERS		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		40			40	
Pitch	p		.100			2.54	
Top to Seating Plane	A	.185	.205	.225	4.70	5.21	5.72
Ceramic Package Height	A2	.155	.160	.165	3.94	4.06	4.19
Standoff	A1	.030	.045	.060	0.76	1.14	1.52
Shoulder to Shoulder Width	E	.595	.600	.625	15.11	15.24	15.88
Ceramic Pkg. Width	E1	.514	.520	.526	13.06	13.21	13.36
Overall Length	D	2.040	2.050	2.060	51.82	52.07	52.32
Tip to Seating Plane	L	.135	.140	.145	3.43	3.56	3.68
Lead Thickness	c	.008	.011	.014	0.20	0.28	0.36
Upper Lead Width	B	.050	.053	.055	1.27	1.33	1.40
Lower Lead Width	B1	.016	.020	.023	0.41	0.51	0.58
Overall Row Spacing §	eB	.610	.660	.710	15.49	16.76	18.03
Window Diameter	W	.340	.350	.360	8.64	8.89	9.14

* Controlling Parameter
§ Significant Characteristic
JEDEC Equivalent: MO-103
Drawing No. C04-014

44-Lead Plastic Thin Quad Flatpack (PT) 10x10x1 mm Body, 1.0/0.10 mm Lead Form (TQFP)

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



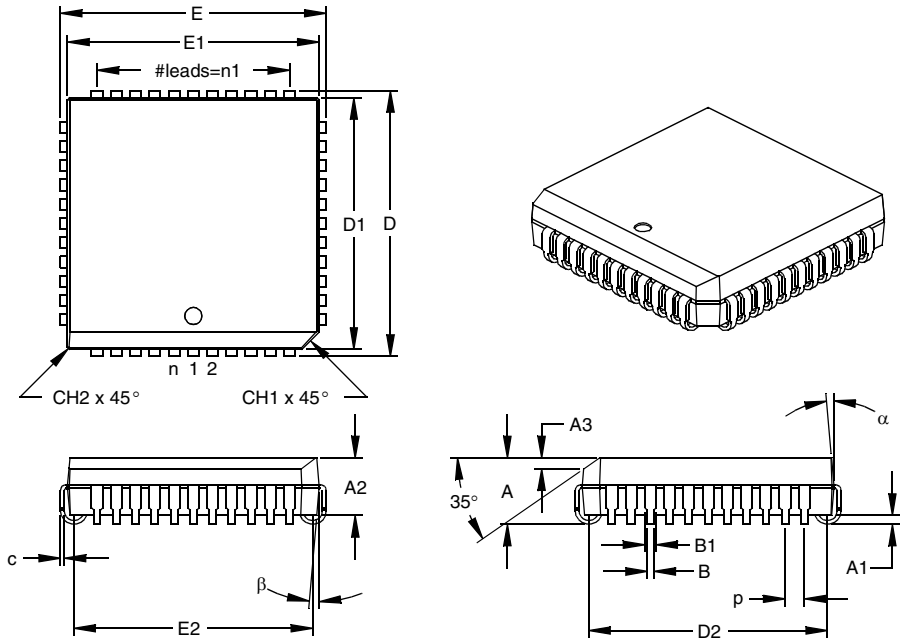
Units		INCHES			MILLIMETERS*		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		44			44	
Pitch	p		.031			0.80	
Pins per Side	n1		11			11	
Overall Height	A	.039	.043	.047	1.00	1.10	1.20
Molded Package Thickness	A2	.037	.039	.041	0.95	1.00	1.05
Standoff §	A1	.002	.004	.006	0.05	0.10	0.15
Foot Length	L	.018	.024	.030	0.45	0.60	0.75
Footprint (Reference)	(F)		.039		1.00		
Foot Angle	φ	0	3.5	7	0	3.5	7
Overall Width	E	.463	.472	.482	11.75	12.00	12.25
Overall Length	D	.463	.472	.482	11.75	12.00	12.25
Molded Package Width	E1	.390	.394	.398	9.90	10.00	10.10
Molded Package Length	D1	.390	.394	.398	9.90	10.00	10.10
Lead Thickness	c	.004	.006	.008	0.09	0.15	0.20
Lead Width	B	.012	.015	.017	0.30	0.38	0.44
Pin 1 Corner Chamfer	CH	.025	.035	.045	0.64	0.89	1.14
Mold Draft Angle Top	α	5	10	15	5	10	15
Mold Draft Angle Bottom	β	5	10	15	5	10	15

* Controlling Parameter
§ Significant Characteristic

Notes:
Dimensions D1 and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010” (0.254mm) per side.
JEDEC Equivalent: MS-026
Drawing No. C04-076

44-Lead Plastic Leaded Chip Carrier (L) – Square (PLCC)

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Units		INCHES*			MILLIMETERS		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		44			44	
Pitch	p		.050			1.27	
Pins per Side	n1		11			11	
Overall Height	A	.165	.173	.180	4.19	4.39	4.57
Molded Package Thickness	A2	.145	.153	.160	3.68	3.87	4.06
Standoff §	A1	.020	.028	.035	0.51	0.71	0.89
Side 1 Chamfer Height	A3	.024	.029	.034	0.61	0.74	0.86
Corner Chamfer 1	CH1	.040	.045	.050	1.02	1.14	1.27
Corner Chamfer (others)	CH2	.000	.005	.010	0.00	0.13	0.25
Overall Width	E	.685	.690	.695	17.40	17.53	17.65
Overall Length	D	.685	.690	.695	17.40	17.53	17.65
Molded Package Width	E1	.650	.653	.656	16.51	16.59	16.66
Molded Package Length	D1	.650	.653	.656	16.51	16.59	16.66
Footprint Width	E2	.590	.620	.630	14.99	15.75	16.00
Footprint Length	D2	.590	.620	.630	14.99	15.75	16.00
Lead Thickness	c	.008	.011	.013	0.20	0.27	0.33
Upper Lead Width	B1	.026	.029	.032	0.66	0.74	0.81
Lower Lead Width	B	.013	.020	.021	0.33	0.51	0.53
Mold Draft Angle Top	α	0	5	10	0	5	10
Mold Draft Angle Bottom	β	0	5	10	0	5	10

* Controlling Parameter
§ Significant Characteristic
Notes:
Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" (0.254mm) per side.
JEDEC Equivalent: MO-047
Drawing No. C04-048

NOTES:

INDEX

A

A/D
- ADCON0 Register 91
- Analog Input Model Block Diagram 95
- Analog-to-Digital Converter 91
- Block Diagram 94
- Configuring Analog Port Pins 96
- Configuring the Interrupt 94
- Configuring the Module 94
- Conversion Clock 96
- Conversions 96
- Converter Characteristics 141
- Effects of a Reset 96
- Faster Conversion - Lower Resolution Tradeoff 96
- Internal Sampling Switch (Rss) Impedance 95
- Operation During Sleep 96
- Sampling Requirements 95
- Source Impedance 95
- Timing Diagram 142
- Using the CCP Trigger 97

Absolute Maximum Ratings 127
ADRES Register 17, 91
Application Notes
- AN552 (Implementing Wake-up on Key Strokes Using PIC16CXXX) 33
- AN556 (Table Reading Using PIC16CXX 29
- AN607, Power-up Trouble Shooting 103

Architecture
- Overview 9

Assembler
- MPASM Assembler 121

B

Baud Rate Formula 79
Block Diagrams
- A/D 94
- Analog Input Model 95
- Capture 53
- Compare 54
- On-Chip Reset Circuit 102
- PORTC 35
- PORTD (In I/O Port Mode) 37
- PORTD and PORTE as a Parallel Slave Port 40
- PORTE (In I/O Port Mode) 38
- PWM 54
- RA4/T0CKI Pin 31
- RB Port Pins 33
- RB Port Pins 33
- Timer0/WDT Prescaler 43
- Timer2 49
- USART Receive 83
- USART Transmit 81
- Watchdog Timer 110

$\overline{\text{BOR}}$ bit 103
BRGH bit 79
Brown-out Reset (BOR)
- Timing Diagram 136

Buffer Descriptor Table 68

C

C bit 22
Capture/Compare/PWM
- Capture
  - Block Diagram 53
  - CCP1CON Register 52
  - CCP1IF 53
  - Mode 53
  - Prescaler 53
- CCP Timer Resources 51
- Compare
  - Block Diagram 54
  - Mode 54
  - Software Interrupt Mode 54
  - Special Event Trigger 54
  - Special Trigger Output of CCP1 54
  - Special Trigger Output of CCP2 54
- Interaction of Two CCP Modules 51
- Section 51
- Special Event Trigger and A/D Conversions 54

Capture/Compare/PWM (CCP)
- PWM Block Diagram 54
- PWM Mode 54
- Timing Diagram 138

CCP1CON 19
CCP2CON 19
CCPR1H Register 17, 19, 51
CCPR1L Register 19, 51
CCPR2H Register 17, 19
CCPR2L Register 17, 19
Clocking Scheme 13
Code Examples
- Call of a Subroutine in Page 1 from Page 0 29
- Changing Prescaler (Timer0 to WDT) 44
- Indirect Addressing 30
- Initializing PORTA 31

Code Protection 99, 112
Computed GOTO 29
Configuration Bits 99
Control 60
CREN bit 78
$\overline{\text{CS}}$ pin 40

D

DC bit 22
DC Characteristics 129, 130
Development Support 5, 121
Direct Addressing 30

E

EC Oscillator 104
Electrical Characteristics 127
Endpoint 71
Errata 3
Error 63

F

FERR bit 78
FSR Register 17, 18, 20, 30

G

General Description 5
GIE bit 107

I

I/O Ports 31
- PORTA 31
- PORTB 33
- PORTC 35
- PORTD 37, 40
- PORTE 38

In-Circuit Serial Programming 99, 112
INDF 19, 20
INDF Register 17, 18, 30
Indirect Addressing 30
Instruction Cycle 13
Instruction Flow/Pipelining 13
Instruction Format 113
Instruction Set
- ADDLW 115
- ADDWF 115
- ANDLW 115
- ANDWF 115
- BCF 115
- BSF 115
- BTFSC 116
- BTFSS 116
- CALL 116
- CLRF 116
- CLRW 116
- CLRWDT 116
- COMF 117
- DECF 117
- DECFSZ 117
- GOTO 117
- INCF 117
- INCFSZ 117
- IORLW 118
- IORWF 118
- MOVF 118
- MOVLW 118
- MOVWF 118
- NOP 118
- RETFIE 119
- RETLW 119
- RETURN 119
- RLF 119
- RRF 119
- SLEEP 119
- SUBLW 1 20
- SUBWF 120
- SWAPF 120
- XORLW 120
- XORWF 120
- Summary Table 114

Instruction Set Summary 113
INT Interrupt 109
INTCON 20
INTCON Register 24
INTEDG bit 109
Internal Sampling Switch (Rss) Impedance 95
Interrupts 99, 107
- PortB Change 109
- RB Port Change 33
- TMR0 109

IRP bit 22

K

KeeLoq® Evaluation and Programming Tools 124

L

Loading of PC 29

M

$\overline{MCLR}$ 101, 104
Memory
- Data Memory 15
- Program Memory 15
- Program Memory Maps
  - PIC16C745/765 15

MPLAB Integrated Development Environment Software 121

O

OERR bit 78
OPCODE 113
OPTION Register 23
OSC selection 99
Oscillator
- E4 100
- EC 100
- H4 100
- HS 100, 104

Oscillator Configurations 100

P

Packaging 147
Paging, Program Memory 29
Parallel Slave Port 37, 40
Parallel Slave Port (PSP)
- Timing Diagram 139

PCL Register 17, 18, 29
PCLATH 105
PCLATH Register 17, 18, 20, 29
PCON Register 28, 103
$\overline{PD}$ bit 22, 101
PICDEM-1 Low-Cost PIC MCU Demo Board 123
PICDEM-2 Low-Cost PIC16CXX Demo Board 123
PICDEM-3 Low-Cost PIC16CXXX Demo Board 123
PICSTART® Plus Entry Level Development System 123
PIE1 Register 25
PIE2 Register 27
Pinout Descriptions
- PIC16C745/765 11

PIR1 Register 26
PIR2 Register 27
POP 29
POR 103
- Oscillator Start-up Timer (OST) 99, 103
- Power Control Register (PCON) 103
- Power-on Reset (POR) 99, 103, 105
- Power-up Timer (PWRT) 99
- Power-Up-Timer (PWRT) 103
- $\overline{TO}$ 101

$\overline{POR}$ bit 103
Port RB Interrupt 109
PORTA 20, 105
PORTA Register 17, 31
PORTB 20, 105
PORTB Register 17, 33
PORTC 20, 105
PORTC Register 17, 35
PORTD 20, 105
PORTD Register 17, 37
PORTE 20, 105
PORTE Register 17, 38
Power-down Mode (SLEEP) 111

Power-on Reset (POR)
Timing Diagram 136
PR2 Register 18, 49
PRO MATE® II Universal Programmer 123
Product Identification System 163
Program Counter
PCLATH Register 109
Program Memory
Paging 29
Program Memory Maps
PIC16C745/765 15
Program Verification 112
PSPMODE bit 37, 38, 40
PUSH 29

R

RBIF bit 33, 109
RCREG 19
RCSTA Register 19, 78
$\overline{RD}$ pin 40
Register File 15
Registers
FSR
Summary 19
INDF
Summary 19
INTCON
Summary 19
PCL
Summary 19
PCLATH
Summary 19
PORTB
Summary 19
Reset Conditions 104
Special Function Register Summary 17
STATUS
Summary 19
TMR0
Summary 19
TRISB
Summary 20
Reset 99, 101
Timing Diagram 136
Reset Conditions for Special Registers 104
RP0 bit 15, 22
RP1 bit 22
RX9 bit 78
RX9D bit 78

S

Serial Communication Interface (SCI) Module, See USART
Services
One-Time-Programmable (OTP) 7
Quick-Turnaround-Production (QTP) 7
Serialized Quick-Turnaround Production (SQTP) 7
SLEEP 99, 101
Software Simulator (MPLAB-SIM) 122
SPBRG Register 18
Special Features of the CPU 99
Special Function Registers 17
PIC16C745/765 17
SPEN bit 78
SREN bit 78
SSPBUF 19
Stack 29
Overflows 29
Underflow 29
Status 64
STATUS Register 22, 109
Synchronous Serial Port Module 57

T

T1CKPS0 bit 45
T1CKPS1 bit 45
T1CON 20
T1CON Register 19, 45
T1OSCEN bit 45
$\overline{T1SYNC}$ bit 45
T2CKPS0 bit 49
T2CKPS1 bit 49
T2CON Register 19, 49
T_{AD} 96
Timer0
RTCC 105
Timing Diagram 137
Timer1
Timing Diagram 137
Timers
Timer0 43
External Clock 44
Interrupt 43
Prescaler 44
Prescaler Block Diagram 43
T0CKI 44
T0IF 109
TMR0 Interrupt 109
Timer1
Asynchronous Counter Mode 47
Capacitor Selection 47
Operation in Timer Mode 46
Oscillator 47
Prescaler 47
Resetting of Timer1 Registers 47
Resetting Timer1 using a CCP
Trigger Output 47
Synchronized Counter Mode 46
T1CON 45
TMR1H 47
TMR1L 47
Timer2
Block Diagram 49
Module 49
Postscaler 49
Prescaler 49
T2CON 49
Timing Diagrams
USART Asynchronous Master Transmission 82
USART Asynchronous Reception 83
USART Synchronous Reception 88
USART Synchronous Transmission 86
Wake-up from Sleep via Interrupt 108, 112
Timing Diagrams and Specifications 133
A/D Conversion 142
Brown-out Reset (BOR) 136
Capture/Compare/PWM (CCP) 138
CLKOUT and I/O 135
External Clock 133
Oscillator Start-up Timer (OST) 136
Parallel Slave Port (PSP) 139
Power-up Timer (PWRT) 136
Reset 136
Timer0 and Timer1 137
USART Synchronous Receive (Master/Slave) 140

USART Synchronous Transmission (Master/Slave) 140
Watchdog Timer (WDT) 136
TMR0 20
TMR0 Register 17
TMR1CS bit 45
TMR1H 20
TMR1H Register 17
TMR1L 20
TMR1L Register 17
TMR1ON bit 45
TMR2 20
TMR2 Register 17
TMR2ON bit 49
$\overline{TO}$ bit 22
TOUTPS0 bit 49
TOUTPS1 bit 49
TOUTPS2 bit 49
TOUTPS3 bit 49
TRISA Register 18, 31
TRISB Register 18, 33
TRISC Register 18, 35
TRISD Register 18, 37
TRISE Register 18, 38, 39
TXREG 19
TXSTA Register 77

U

Universal Synchronous Asynchronous Receiver Transmitter (USART) 77
USART
Asynchronous Mode 81
Asynchronous Receiver 83
Asynchronous Reception 84
Asynchronous Transmitter 81
Baud Rate Generator (BRG) 79
Receive Block Diagram 83
Sampling 79
Synchronous Master Mode 85
Timing Diagram, Synchronous Receive 140
Timing Diagram, Synchronous Transmission 140
Synchronous Master Reception 87
Synchronous Master Transmission 85
Synchronous Slave Mode 89
Synchronous Slave Reception 89
Synchronous Slave Transmit 89
Transmit Block Diagram 81
USB 21, 58, 60, 61, 62
USB Address Register 66
USB Control Register 65
USB Endpoint Control Register 67
UV Erasable Devices 7

W

W Register 109
Wake-up from SLEEP 111
Watchdog Timer (WDT) 99, 101, 104, 110
Timing Diagram 136
WDT 104
Block Diagram 110
Period 110
Programming Considerations 110
Timeout 105
$\overline{WR}$ pin 40
WWW, On-Line Support 3

Z

Z bit 22

THE MICROCHIP WEB SITE

Microchip provides online support via our WWW site at www.microchip.com. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

CUSTOMER CHANGE NOTIFICATION SERVICE

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip web site at www.microchip.com. Under "Support", click on "Customer Change Notification" and follow the registration instructions.

CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or field application engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the web site at: http://microchip.com/support

READER RESPONSE

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

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

TO: Technical Publications Manager Total Pages Sent _____

RE: Reader Response

From: Name _____

Company _____

Address _____

City / State / ZIP / Country _____

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

Application (optional):

Would you like a reply? ___ Y ___ N

Device: Literature Number: DS41124D

Questions:

1. What are the best features of this document?

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

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

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

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

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

7. How would you improve this document?

PRODUCT IDENTIFICATION SYSTEM

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

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

Device	PIC16C745[1], PIC16C745T[2] PIC16C765[1], PIC16C765T[2]	
Temperature Range	I	= -40°C to +85°C (Industrial)
Package	JW PT SO SP P L	= Windowed CERDIP - 600 mil = TQFP (Thin Quad Flatpack) = SOIC = Skinny plastic dip = PDIP = PLCC
Pattern	QTP Code or Special Requirements (blank otherwise)	

Examples:

a) PIC16C745-I/P 301 = Industrial temp., PDIP package, QTP pattern #301.

Note 1: C = CMOS
Note 2: T = in tape and reel - SOIC, PLCC, TQFP, packages only.

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

Sales and Support

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

1. Your local Microchip sales office
2. The Microchip Worldwide Site (www.microchip.com)

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Trademarks

Printed on recycled paper.

ISBN: 9781620769690

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
═ ISO/TS 16949 ═

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Cleveland
Independence, OH
Tel: 216-447-0464
Fax: 216-447-0643

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Farmington Hills, MI
Tel: 248-538-2250
Fax: 248-538-2260

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

Santa Clara
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

Toronto
Mississauga, Ontario, Canada
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Chongqing
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

China - Hangzhou
Tel: 86-571-2819-3187
Fax: 86-571-2819-3189

China - Hong Kong SAR
Tel: 852-2943-5100
Fax: 852-2401-3431

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8864-2200
Fax: 86-755-8203-1760

China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

Japan - Osaka
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310

Japan - Tokyo
Tel: 81-3-6880- 3770
Fax: 81-3-6880-3771

Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu
Tel: 886-3-5778-366
Fax: 886-3-5770-955

Taiwan - Kaohsiung
Tel: 886-7-213-7828
Fax: 886-7-330-9305

Taiwan - Taipei
Tel: 886-2-2508-8600
Fax: 886-2-2508-0102

Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

UK - Wokingham
Tel: 44-118-921-5869
Fax: 44-118-921-5820

11/29/12