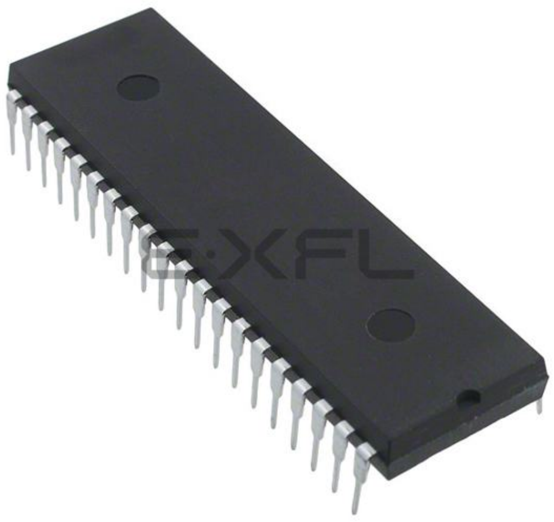


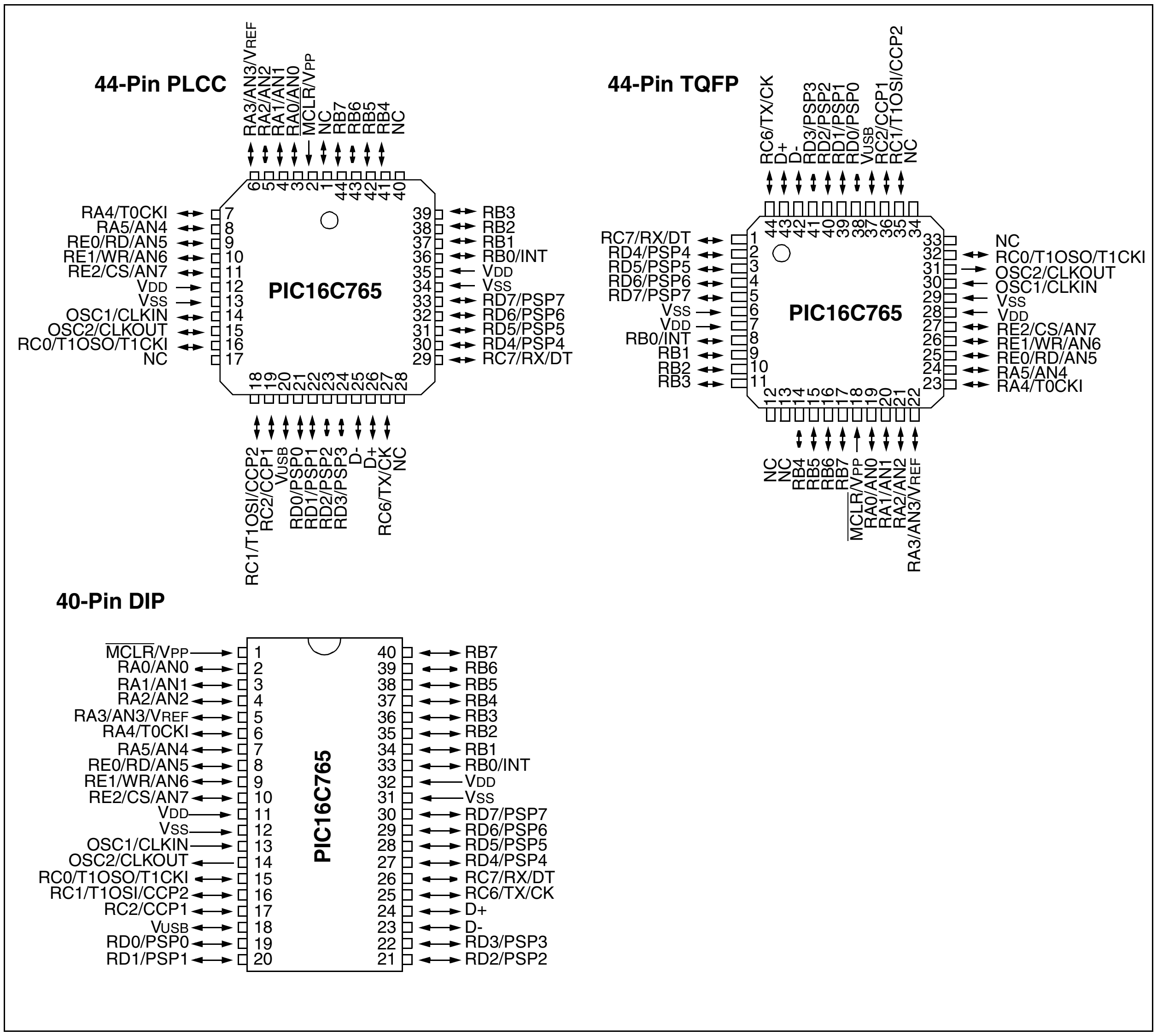
Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	24MHz
Connectivity	SCI, UART/USART, USB
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	33
Program Memory Size	14KB (8K x 14)
Program Memory Type	OTP
EEPROM Size	-
RAM Size	256 x 8
Voltage - Supply (Vcc/Vdd)	4.35V ~ 5.25V
Data Converters	A/D 8x8b
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Through Hole
Package / Case	40-DIP (0.600", 15.24mm)
Supplier Device Package	40-PDIP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic16c765-i-p



Key Features PICmicro^{TM} 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

Preliminary

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/$\overline{\text{RD}}$/AN5	RE0	ST	CMOS	Bi-directional I/O(2)
	$\overline{\text{RD}}$	TTL	—	Parallel Slave Port Control Input(2)
	AN5	AN	—	A/D Input(2)
RE1/$\overline{\text{WR}}$/AN6	RE1	ST	CMOS	Bi-directional I/O(2)
	$\overline{\text{WR}}$	TTL	—	Parallel Slave Port Control Input(2)
	AN6	AN	—	A/D Input(2)
RE2/$\overline{\text{CS}}$/AN7	RE2	ST	CMOS	Bi-directional I/O(2)
	$\overline{\text{CS}}$	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.

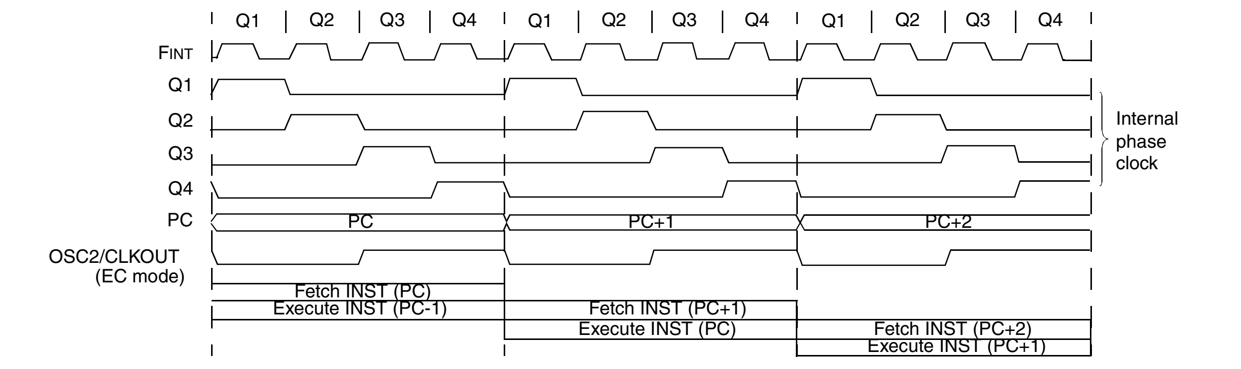
3.2 Instruction Flow/Pipelining

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

	$T_{CY}0$	$T_{CY}1$	$T_{CY}2$	$T_{CY}3$	$T_{CY}4$	$T_{CY}5$
`1. MOVLW 55h`	Fetch 1	Execute 1				
`2. MOVWF PORTB`		Fetch 2	Execute 2			
`3. CALL  SUB_1`			Fetch 3	Execute 3		
`4. BSF   PORTA, BIT3 (Forced NOP)`				Fetch 4	Flush	
`5. Instruction @ address SUB_1`					Fetch SUB_1	Execute SUB_1

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.

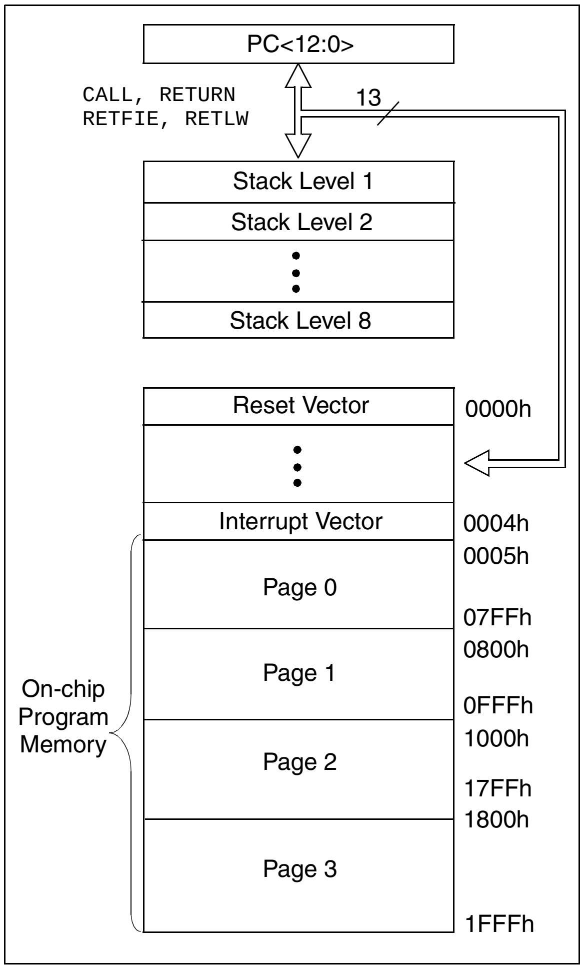
4.0 MEMORY ORGANIZATION

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).

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.

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

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.

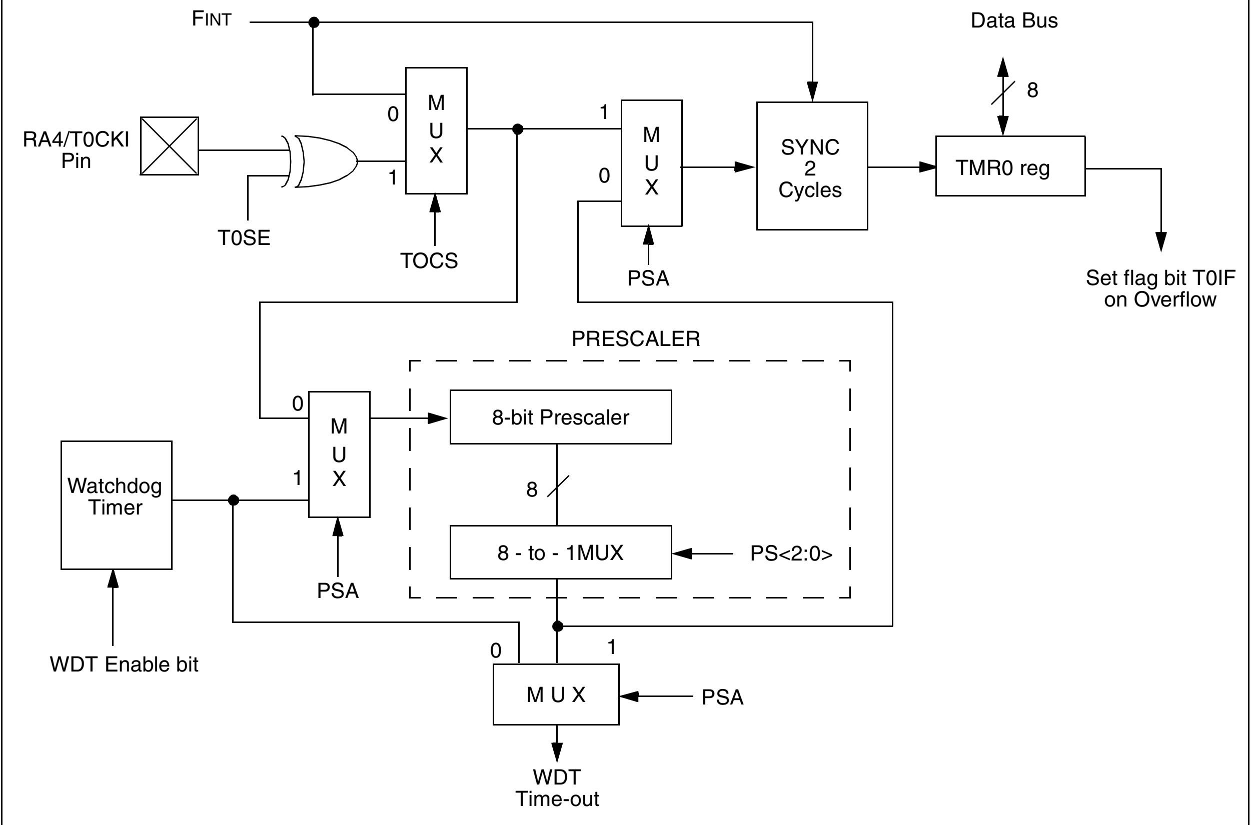
Counter mode is selected by setting bit T0CS (OPTION_REG<5>). In counter mode, Timer0 will increment either on every rising or falling edge of pin RA4/T0CKI. The incrementing edge is determined by the Timer0 Source Edge Select bit T0SE (OPTION_REG<4>). Clearing bit T0SE selects the rising edge. Restrictions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

7.3 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.

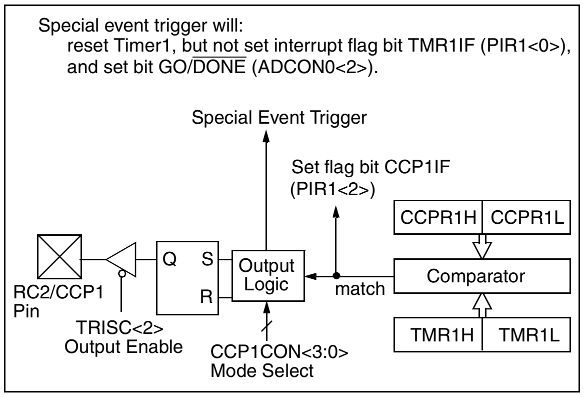
9.2 Compare Mode

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


Special event trigger will:
reset Timer1, but not set interrupt flag bit TMR1IF (PIR1<0>), and set bit GO/DONE (ADCON0<2>).
Special Event Trigger
Set flag bit CCP1IF (PIR1<2>)
CCPR1H
CCPR1L
Q
S
R
Output Logic
match
Comparator
RC2/CCP1 Pin
TRISC<2> Output Enable
CCP1CON<3:0> Mode Select
TMR1H
TMR1L

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)

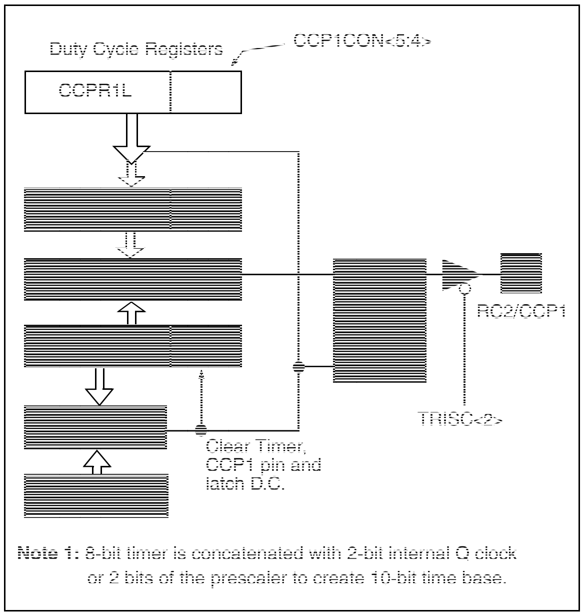
In pulse width modulation mode, the CCPx pin produces up to a 10-bit resolution PWM output. Since the CCP1 pin is multiplexed with the PORTC data latch, the TRISC<2> bit must be cleared to make the CCP1 pin an output.

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


Duty Cycle Registers
CCP1CON<5:4>
CCPR1L
RC2/CCP1
TRISC<2>
Clear Timer, CCP1 pin and latch D.C.
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).

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" ($\leq$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.

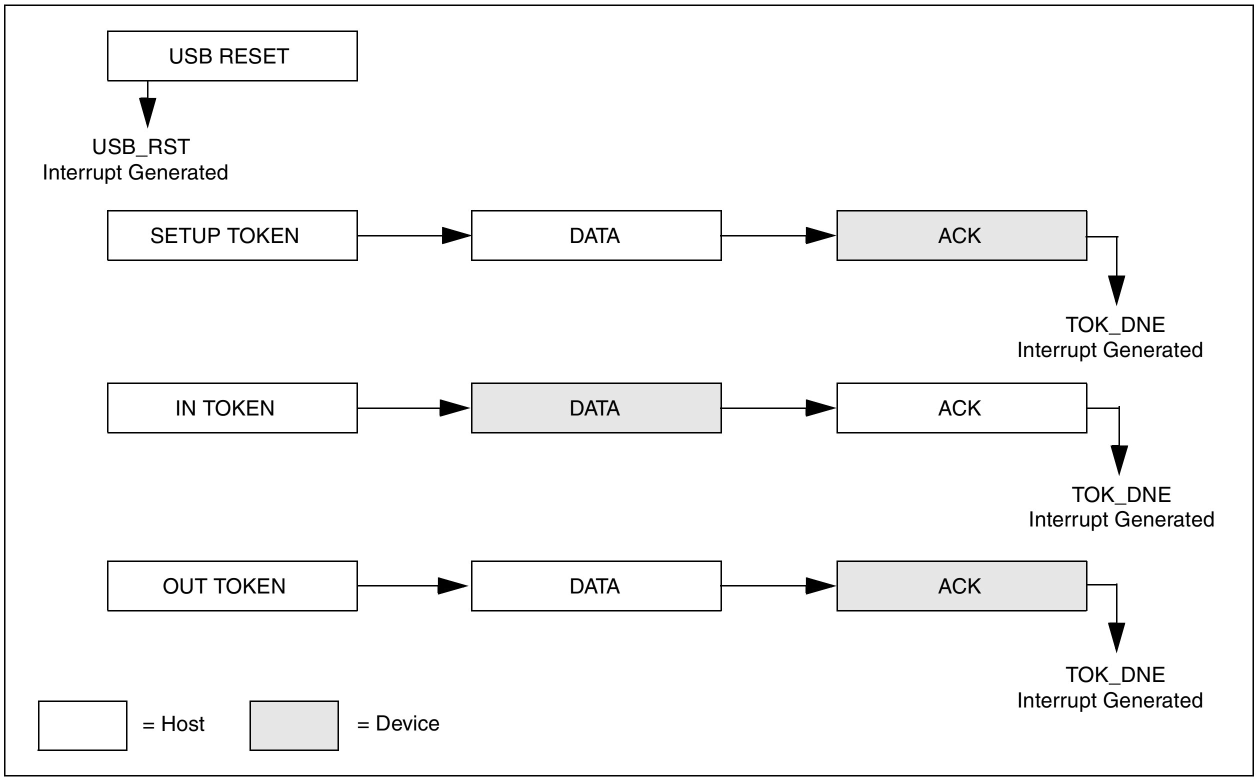
10.1.5 ENUMERATION

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.

FIGURE 10-1: USB TOKENS



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

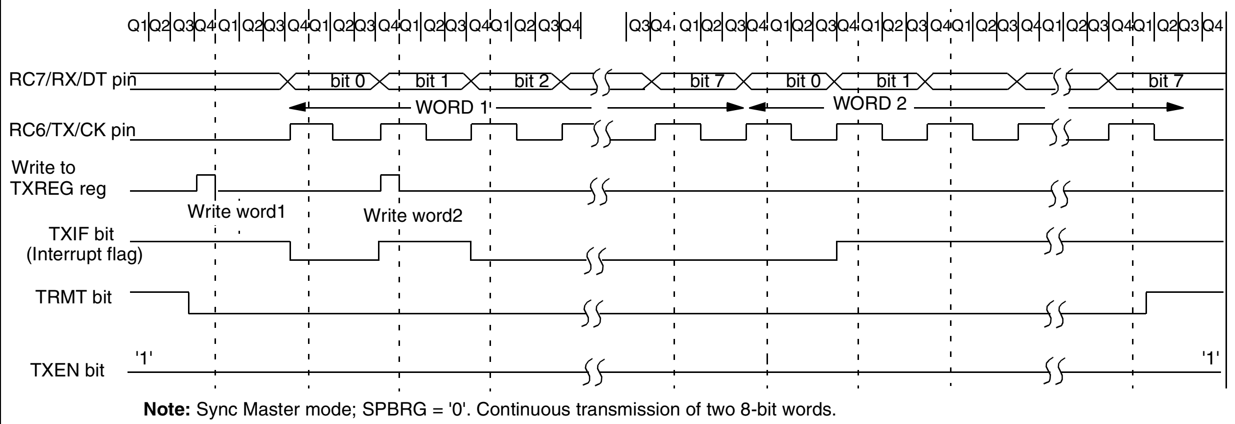
bit 1-0: **Unimplemented:** Read as ’0’

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

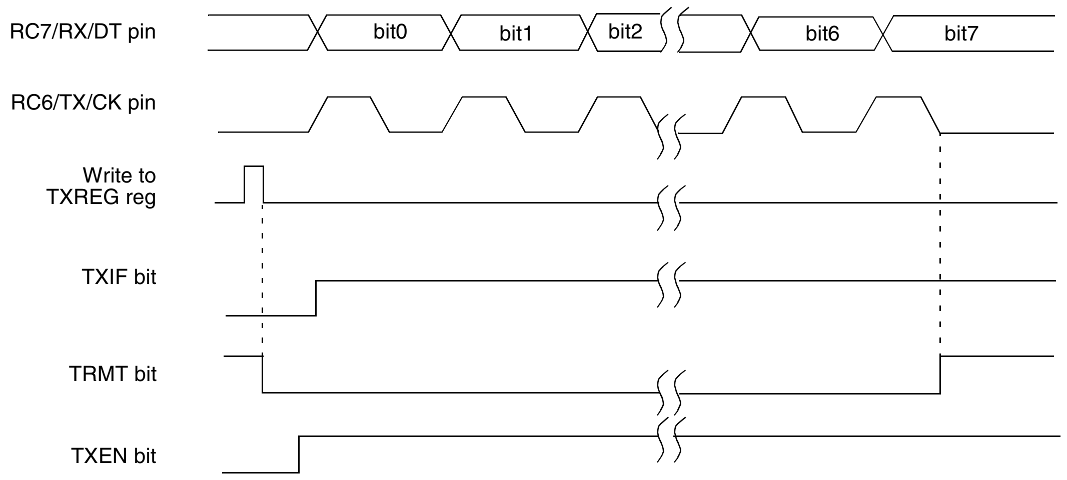
Legend: x = unknown, - = unimplemented, read as '0'. Shaded cells are not used for synchronous master transmission.
Note 1: Bits PSPIE and PSPIF are reserved on the PIC16C745; always maintain these bits clear.

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)



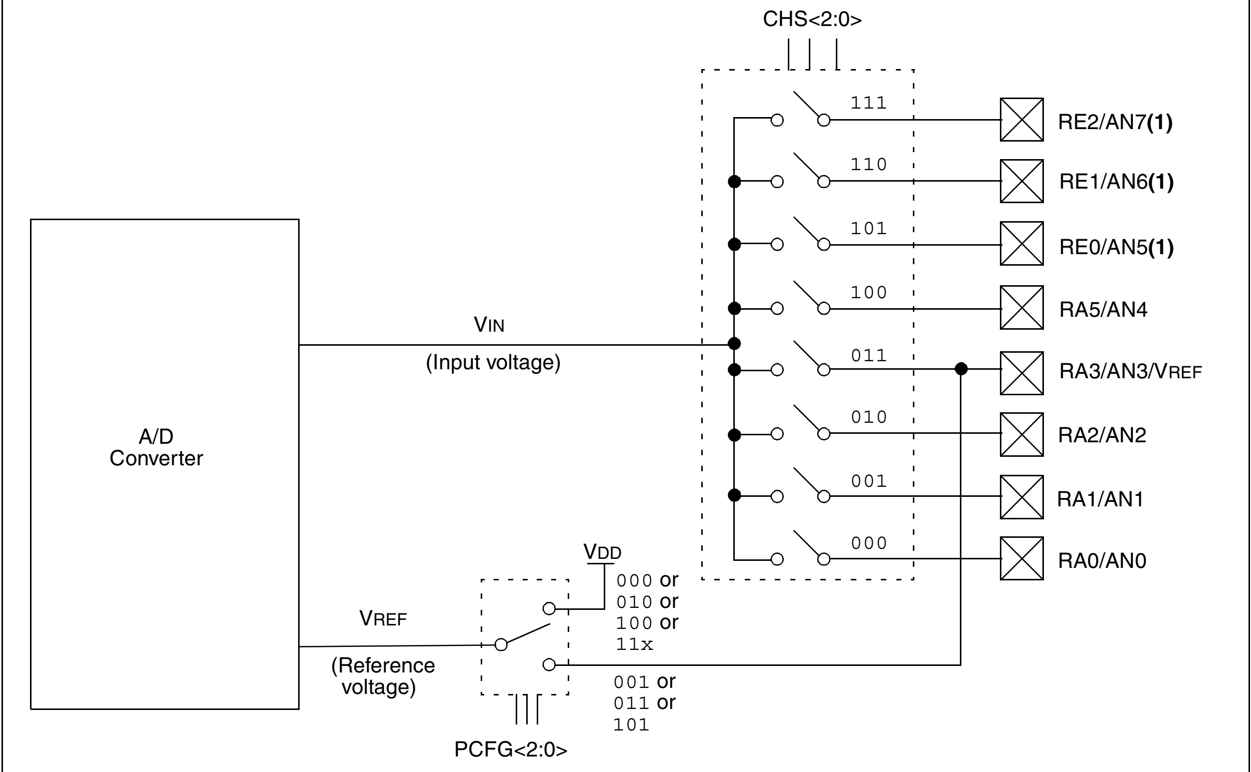
The following steps should be followed for doing an A/D conversion:

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

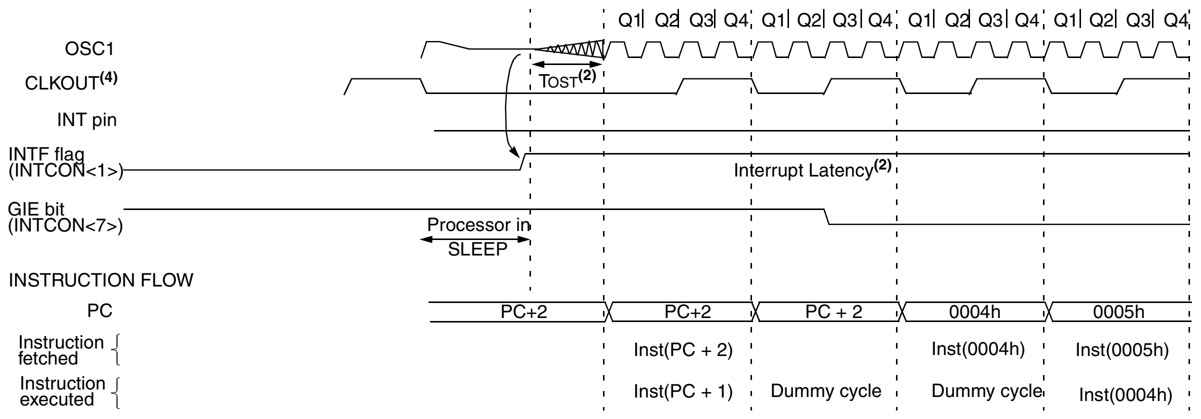
   - Waiting for the A/D interrupt
6. Read A/D result register (ADRES), clear bit ADIF if required.
7. For next conversion, go to step 1 or step 2 as required. The A/D conversion time per bit is defined as TAD. A minimum wait of 2TAD is required before next acquisition starts.

FIGURE 12-1: A/D BLOCK DIAGRAM



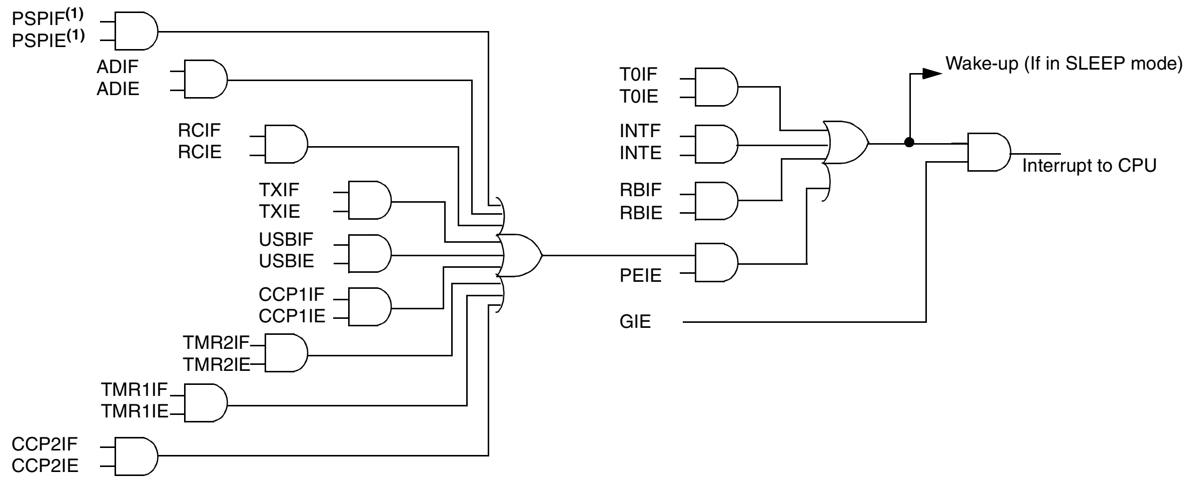
Note 1: Not available on PIC16C745.

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.

IORLW	**Inclusive OR Literal with W**
Syntax:	[*label*] IORLW k
Operands:	$0 \leq k \leq 255$
Operation:	(W) .OR. k $\rightarrow$ (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:	$0 \leq f \leq 127$ $d \in [0,1]$
Operation:	(W) .OR. (f) $\rightarrow$ (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:	$0 \leq f \leq 127$ $d \in [0,1]$
Operation:	(f) $\rightarrow$ (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:	$0 \leq k \leq 255$
Operation:	k $\rightarrow$ (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:	$0 \leq f \leq 127$
Operation:	(W) $\rightarrow$ (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.

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	μA	During VAIN acquisition. Based on differential of VHOLD to VAIN to charge CHOLD, see Section 12.1.
			—	—	10	μA	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.

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)

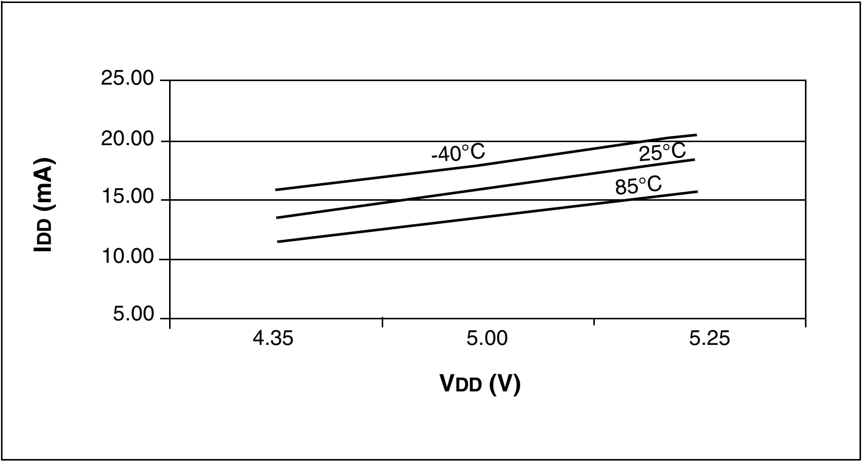
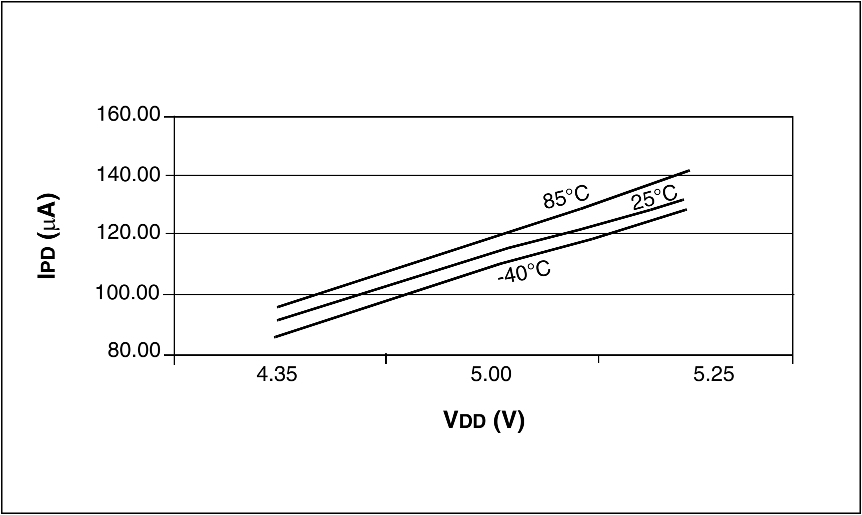


FIGURE 17-2: TYPICAL IPD vs. VDD (USB SUSPENDED, WDT DISABLED)



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