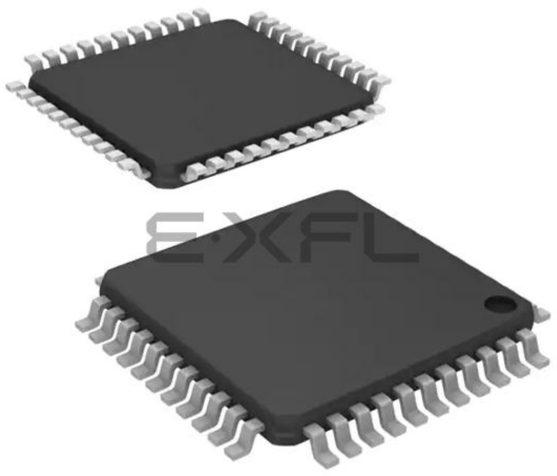


Welcome to E-XFL.COM

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	Surface Mount
Package / Case	44-TQFP
Supplier Device Package	44-TQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic16c765t-i-pt

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.

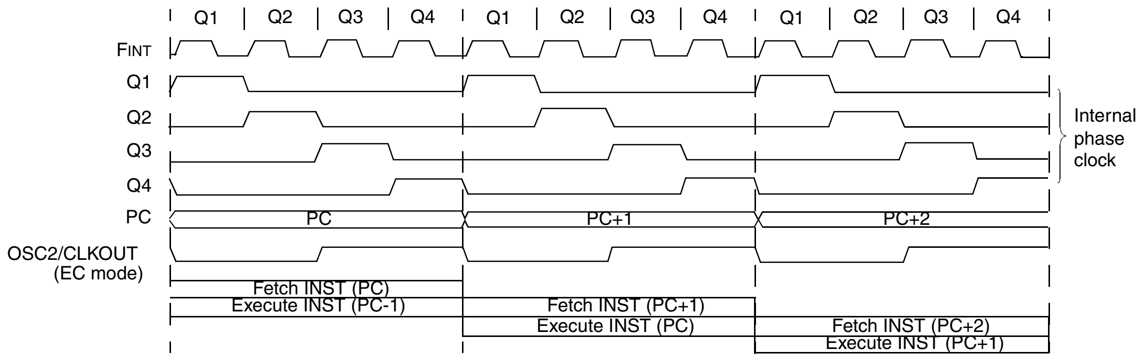
3.2 Instruction Flow/Pipelining

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

	TCY0	TCY1	TCY2	TCY3	TCY4	TCY5
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.

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.

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.

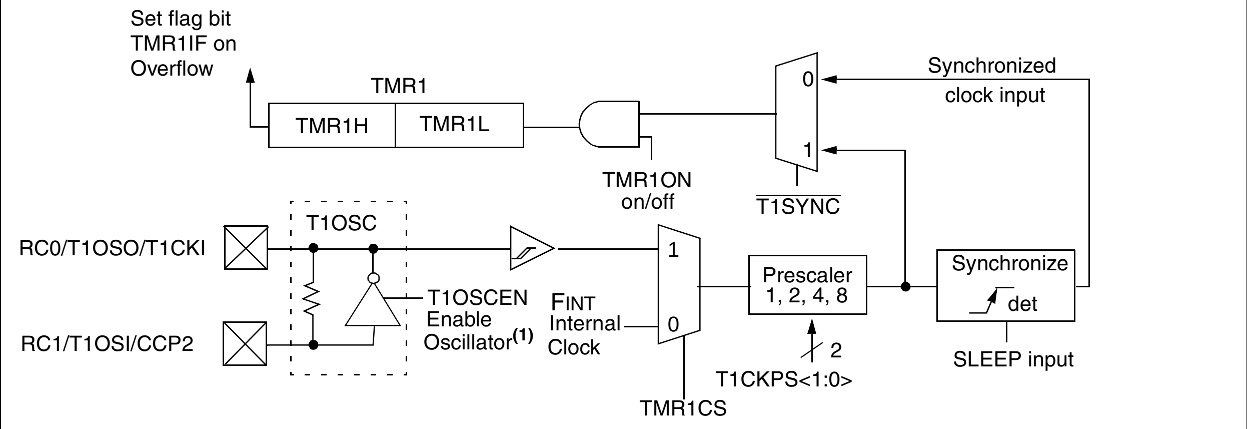
7.2 Timer1 Operation in Synchronized Counter Mode

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.

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.

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.

NOTES:

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.)

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

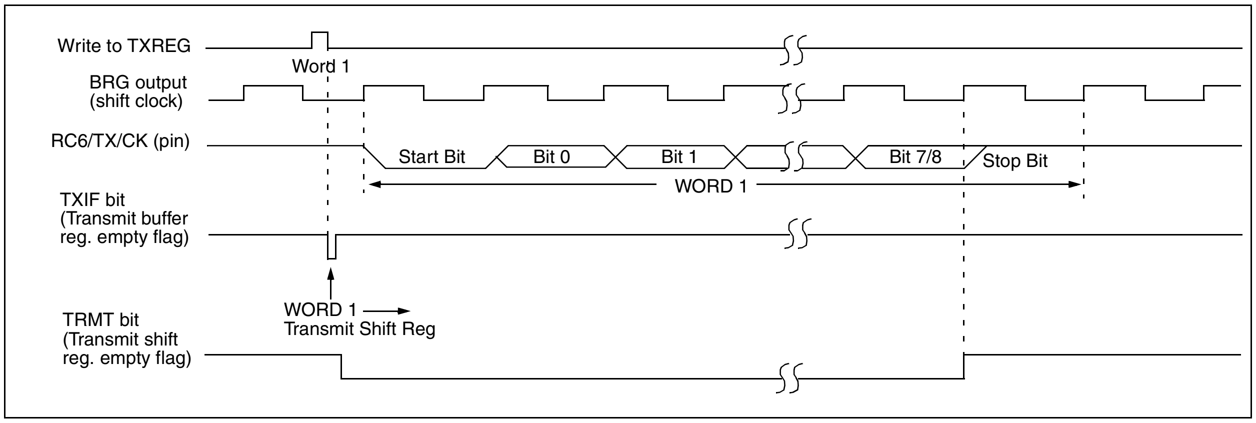
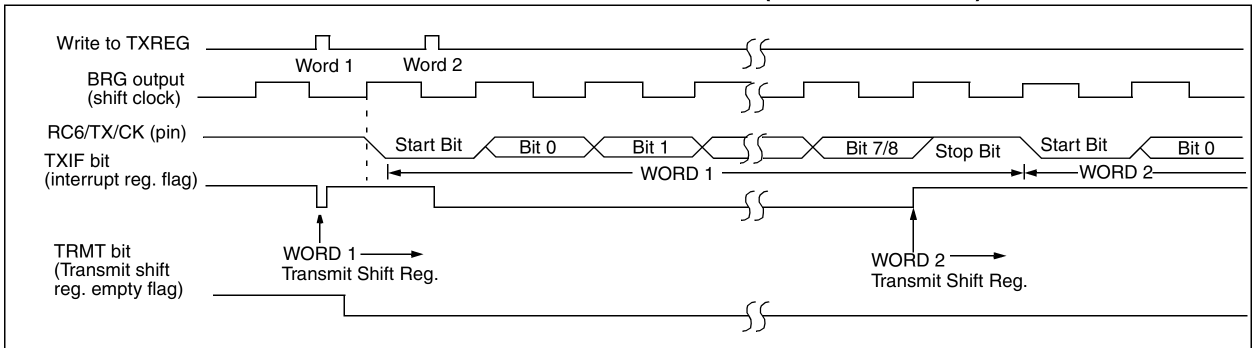

Write to TXREG
Word 1
BRG output (shift clock)
RC6/TX/CK (pin)
Start Bit
Bit 0
Bit 1
Bit 7/8
Stop Bit
WORD 1
TXIF bit (Transmit buffer reg. empty flag)
WORD 1 → Transmit Shift Reg
TRMT bit (Transmit shift reg. empty flag)

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


Write to TXREG
Word 1
Word 2
BRG output (shift clock)
RC6/TX/CK (pin)
Start Bit
Bit 0
Bit 1
Bit 7/8
Stop Bit
Start Bit
Bit 0
TXIF bit (interrupt reg. flag)
WORD 1
WORD 2
WORD 1 → Transmit Shift Reg.
WORD 2 → Transmit Shift Reg.
TRMT bit (Transmit shift reg. empty flag)

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.

11.4 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.

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 $\overline{\text{MCLR}}$ 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 $\overline{\text{PWRTE}}$ = 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 $\overline{\text{MCLR}}$ is kept low long enough, the time-outs will expire. Bringing $\overline{\text{MCLR}}$ high will begin execution immediately. This is useful for testing purposes or to synchronize more than one PIC16CXX device operating in parallel.

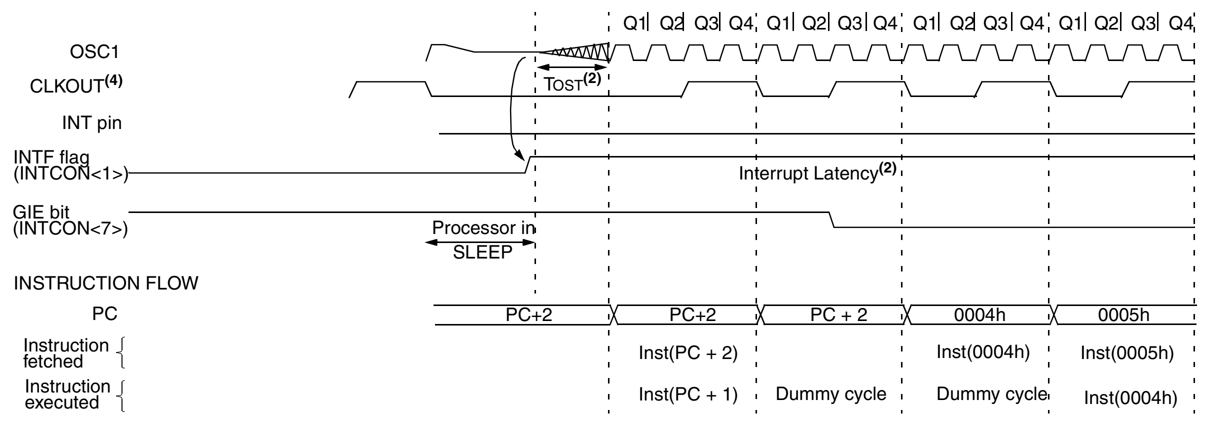
Table 13-5 shows the RESET conditions for the STATUS, PCON and PC registers, while Table 13-7 shows the RESET conditions for all the registers.

13.4.6 POWER CONTROL/STATUS REGISTER (PCON)

The Brown-out Reset Status bit, $\overline{\text{BOR}}$, is unknown on a POR. It must be set by the user and checked on subsequent RESETS to see if bit $\overline{\text{BOR}}$ was cleared, indicating a BOR occurred. The $\overline{\text{BOR}}$ bit is not predictable if the Brown-out Reset circuitry is disabled.

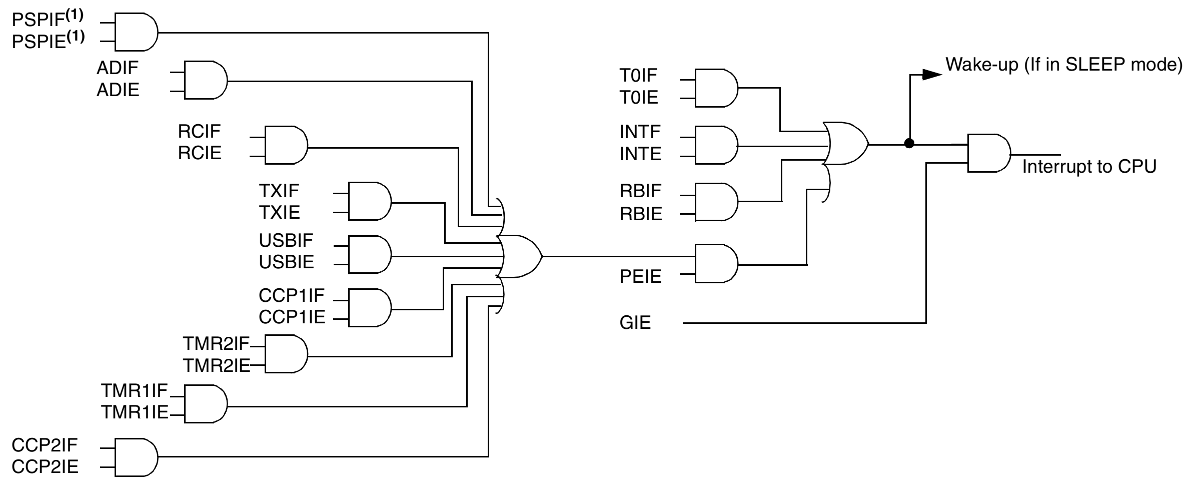
The Power-on Reset Status bit, $\overline{\text{POR}}$, is cleared on a POR and unaffected otherwise. The user must set this bit following a POR and check it on subsequent RESETS to see if it has been cleared.

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 $\rightarrow$ 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
```

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
$\overline{\text{TO}}$	Time-out bit
$\overline{\text{PD}}$	Power-down bit
`dest`	Destination either the W register or the specified register file location
[ ]	Options
( )	Contents
$\rightarrow$	Assigned to
< >	Register bit field
$\in$	In the set of
italics	User defined term (font is courier)

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

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

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

Table 14-2 lists the instructions recognized by the MPASM assembler.

Figure 14-1 shows the general formats that the instructions can have.

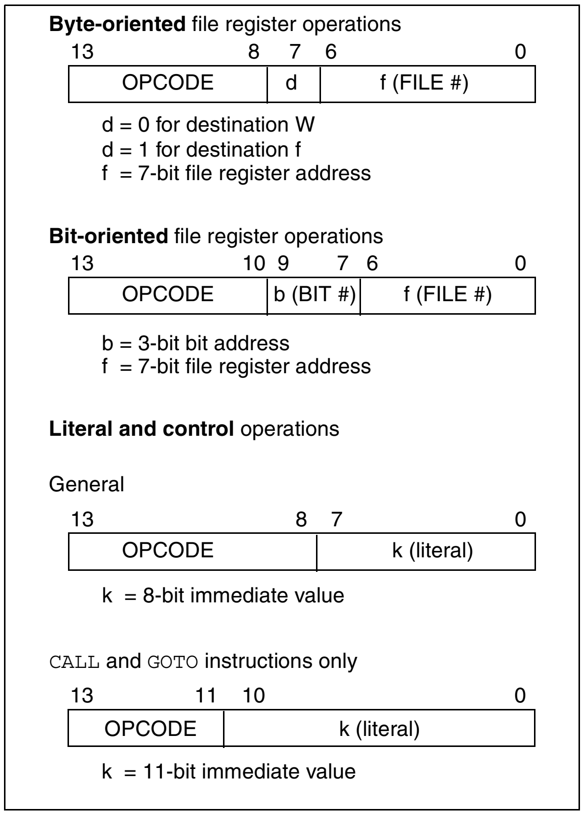
Note: To maintain upward compatibility with future PIC16CXX products, do not use the `OPTION` and `TRIS` instructions.

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


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

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'.

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'.

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.

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.

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'.

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.

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.

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 = $V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD}-V_{OH}) \times I_{OH}\} + \sum(V_{Ol} \times I_{OL})$

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.

16.3.3 TIMING DIAGRAMS AND SPECIFICATIONS

FIGURE 16-3: EXTERNAL CLOCK TIMING

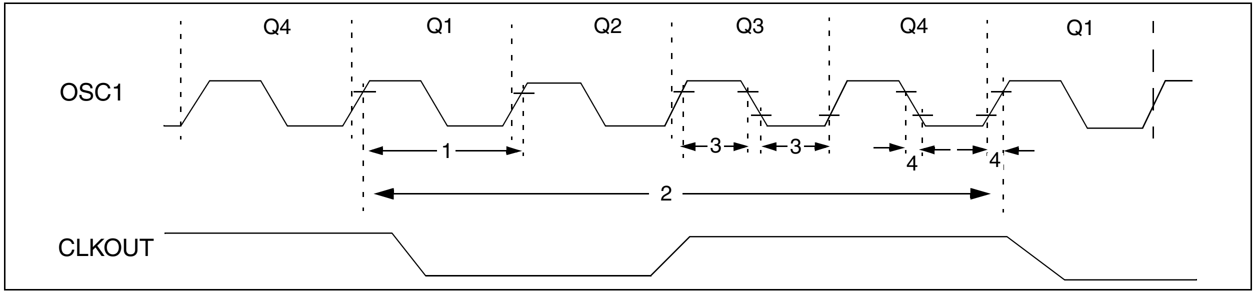
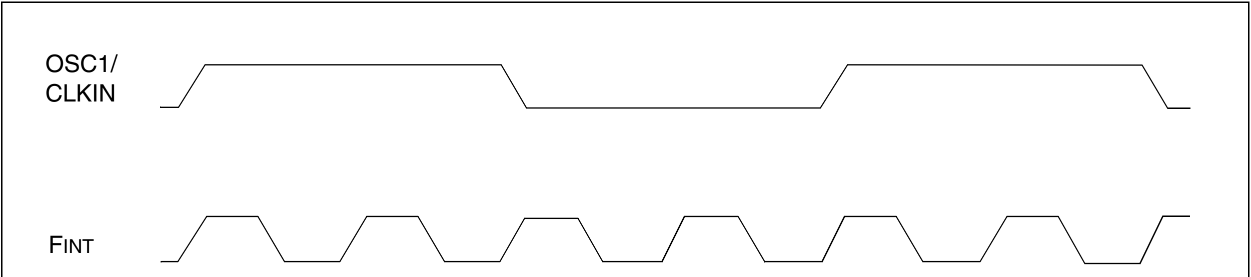


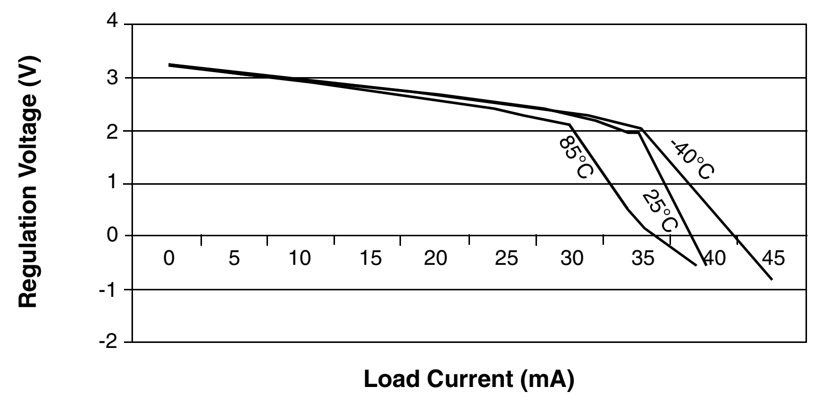
FIGURE 16-4: CLOCK MULTIPLIER (PLL) PHASE RELATIONSHIP



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.

FIGURE 17-3: DC LOAD LINES FOR USB REGULATOR OUTPUT (VUSB)



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