



Welcome to [E-XFL.COM](https://www.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	Obsolete
Core Processor	PIC
Core Size	8-Bit
Speed	25MHz
Connectivity	UART/USART
Peripherals	POR, PWM, WDT
Number of I/O	33
Program Memory Size	16KB (8K x 16)
Program Memory Type	OTP
EEPROM Size	-
RAM Size	454 x 8
Voltage - Supply (Vcc/Vdd)	4.5V ~ 6V
Data Converters	-
Oscillator Type	External
Operating Temperature	-40°C ~ 125°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/pic17c44-25e-pt

PIC17C4X

NOTES:

3.1 Clocking Scheme/Instruction Cycle

The clock input (from OSC1) 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, and 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 are shown in Figure 3-3.

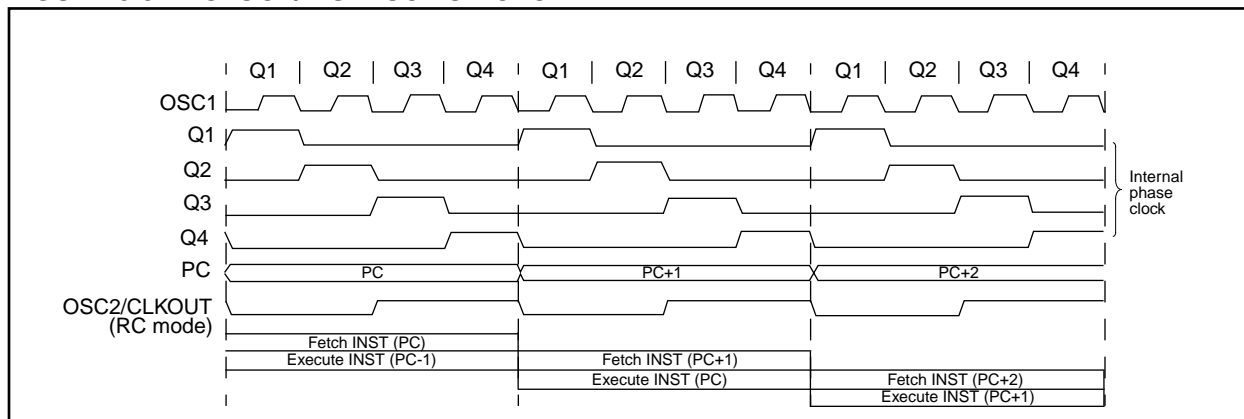
3.2 Instruction Flow/Pipelining

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 3-2).

A fetch cycle begins with the program counter 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 3-3: CLOCK/INSTRUCTION CYCLE



EXAMPLE 3-2: INSTRUCTION PIPELINE FLOW

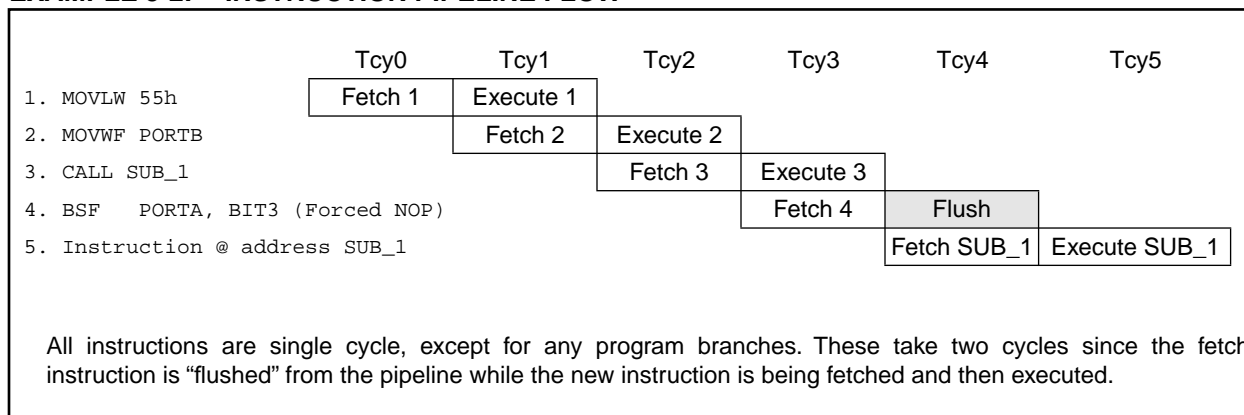


FIGURE 4-2: TIME-OUT SEQUENCE ON POWER-UP ($\overline{\text{MCLR}}$ TIED TO V_{DD})

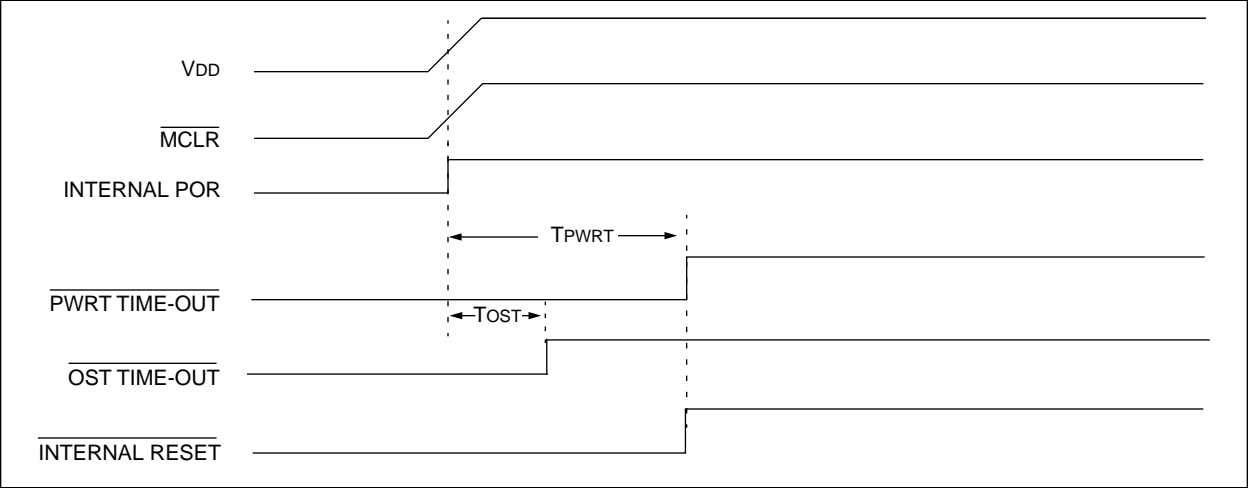


FIGURE 4-3: TIME-OUT SEQUENCE ON POWER-UP ($\overline{\text{MCLR}}$ NOT TIED TO V_{DD})

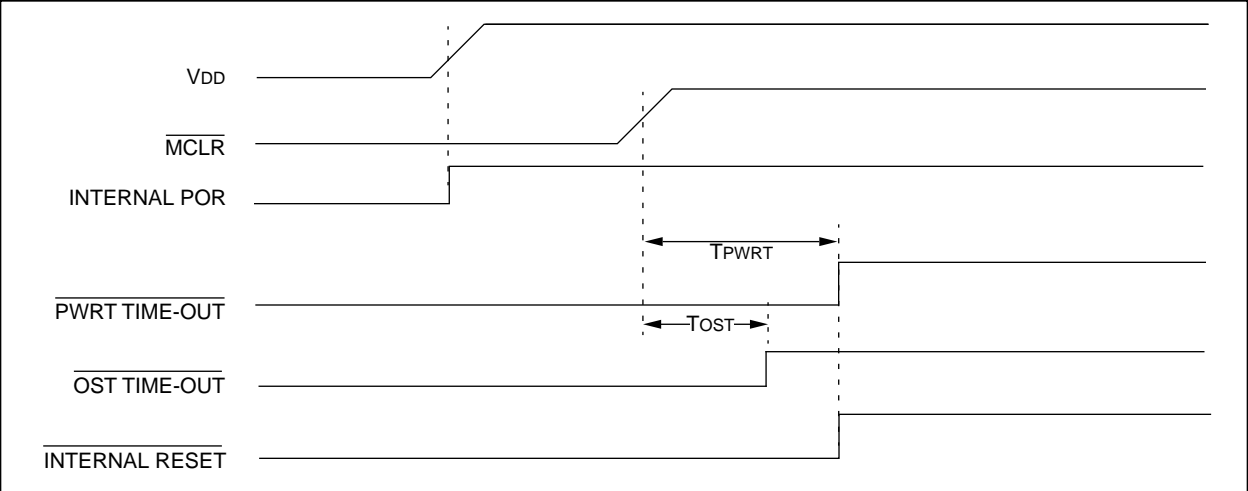
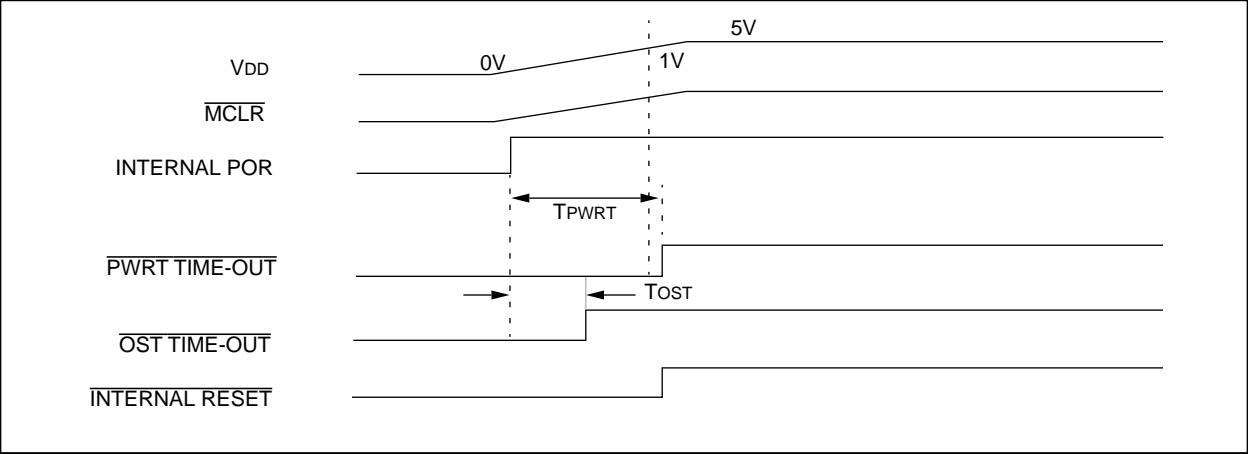


FIGURE 4-4: SLOW RISE TIME ($\overline{\text{MCLR}}$ TIED TO V_{DD})



5.9 Context Saving During Interrupts

During an interrupt, only the returned PC value is saved on the stack. Typically, users may wish to save key registers during an interrupt; e.g. WREG, ALUSTA and the BSR registers. This requires implementation in software.

Example 5-1 shows the saving and restoring of information for an interrupt service routine. The PUSH and POP routines could either be in each interrupt service routine or could be subroutines that were called. Depending on the application, other registers may also need to be saved, such as PCLATH.

EXAMPLE 5-1: SAVING STATUS AND WREG IN RAM

```
;
; The addresses that are used to store the CPUTA and WREG values
; must be in the data memory address range of 18h - 1Fh. Up to
; 8 locations can be saved and restored using
; the MOVFP instruction. This instruction neither affects the status
; bits, nor corrupts the WREG register.
;
;
PUSH    MOVFP    WREG, TEMP_W           ; Save WREG
        MOVFP    ALUSTA, TEMP_STATUS   ; Save ALUSTA
        MOVFP    BSR, TEMP_BSR         ; Save BSR

ISR     :
        :                               ; This is the interrupt service routine
        :
POP     MOVFP    TEMP_W, WREG           ; Restore WREG
        MOVFP    TEMP_STATUS, ALUSTA   ; Restore ALUSTA
        MOVFP    TEMP_BSR, BSR         ; Restore BSR
        RETFIE                          ; Return from Interrupts enabled
```

PIC17C4X

6.2.2.3 TMR0 STATUS/CONTROL REGISTER (T0STA)

This register contains various control bits. Bit7 (INTEDG) is used to control the edge upon which a signal on the RA0/INT pin will set the RB0/INT interrupt flag. The other bits configure the Timer0 prescaler and clock source. (Figure 11-1).

FIGURE 6-9: T0STA REGISTER (ADDRESS: 05h, UNBANKED)

R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	U - 0
INTEDG	T0SE	T0CS	PS3	PS2	PS1	PS0	—
bit7							bit0

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

bit 7: **INTEDG:** RA0/INT Pin Interrupt Edge Select bit
This bit selects the edge upon which the interrupt is detected.
1 = Rising edge of RA0/INT pin generates interrupt
0 = Falling edge of RA0/INT pin generates interrupt

bit 6: **T0SE:** Timer0 Clock Input Edge Select bit
This bit selects the edge upon which TMR0 will increment.
When T0CS = 0
1 = Rising edge of RA1/T0CKI pin increments TMR0 and/or generates a T0CKIF interrupt
0 = Falling edge of RA1/T0CKI pin increments TMR0 and/or generates a T0CKIF interrupt
When T0CS = 1
Don't care

bit 5: **T0CS:** Timer0 Clock Source Select bit
This bit selects the clock source for Timer0.
1 = Internal instruction clock cycle (TCY)
0 = T0CKI pin

bit 4-1: **PS3:PS0:** Timer0 Prescale Selection bits
These bits select the prescale value for Timer0.

PS3:PS0	Prescale Value
0000	1:1
0001	1:2
0010	1:4
0011	1:8
0100	1:16
0101	1:32
0110	1:64
0111	1:128
1xxx	1:256

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

7.0 TABLE READS AND TABLE WRITES

The PIC17C4X has four instructions that allow the processor to move data from the data memory space to the program memory space, and vice versa. Since the program memory space is 16-bits wide and the data memory space is 8-bits wide, two operations are required to move 16-bit values to/from the data memory.

The `TLWT t,f` and `TABLWT t,i,f` instructions are used to write data from the data memory space to the program memory space. The `TLRD t,f` and `TABLRD t,i,f` instructions are used to write data from the program memory space to the data memory space.

The program memory can be internal or external. For the program memory access to be external, the device needs to be operating in extended microcontroller or microprocessor mode.

Figure 7-1 through Figure 7-4 show the operation of these four instructions.

FIGURE 7-1: TLWT INSTRUCTION OPERATION

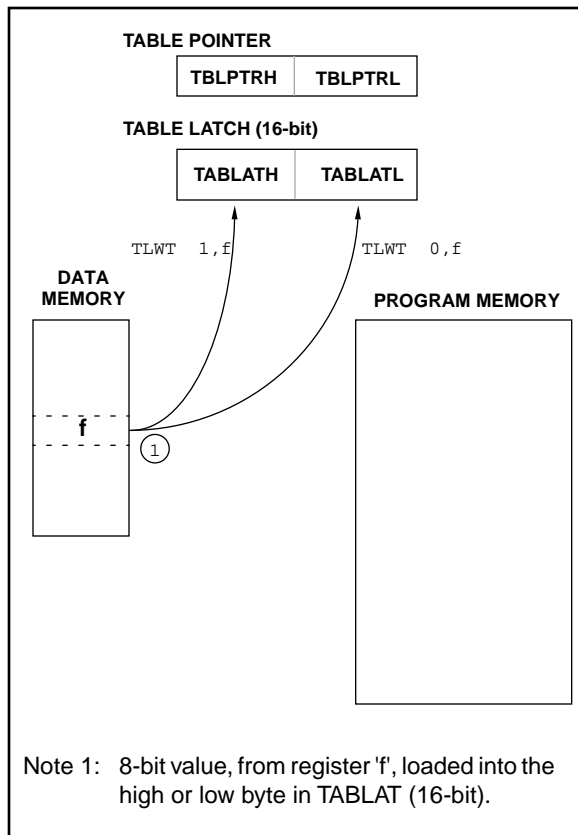
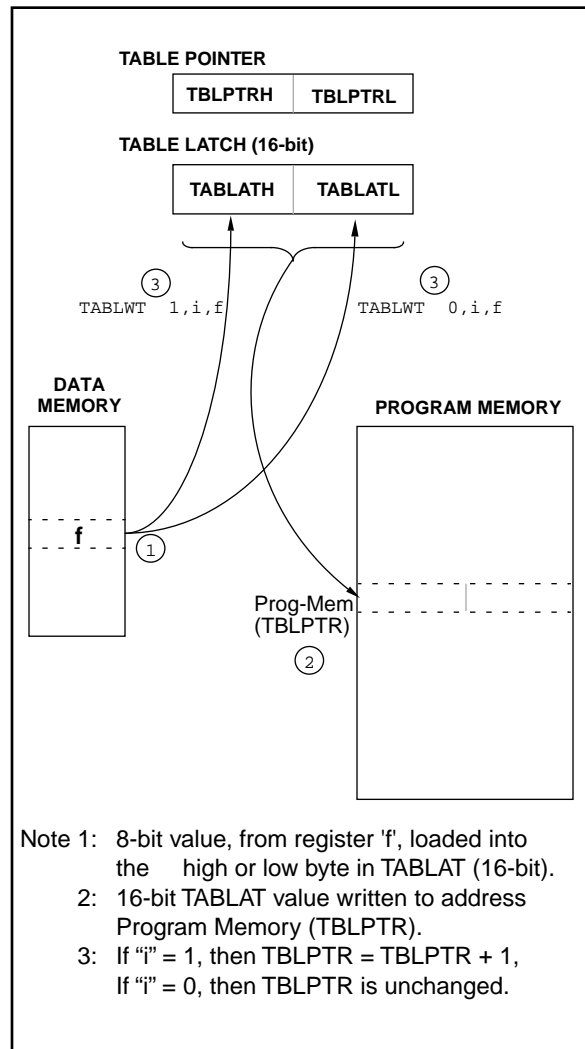


FIGURE 7-2: TABLWT INSTRUCTION OPERATION



9.5 I/O Programming Considerations

9.5.1 BI-DIRECTIONAL I/O PORTS

Any instruction which writes, operates internally as a read followed by a write operation. For example, the BCF and BSF instructions read the register into the CPU, execute the bit operation, and write the result back to the register. Caution must be used when these instructions are applied to a port with both inputs and outputs defined. For example, a BSF operation on bit5 of PORTB will cause all eight bits of PORTB to be read into the CPU. Then the BSF operation takes place on bit5 and PORTB is written to the output latches. If another bit of PORTB is used as a bi-directional I/O pin (e.g. bit0) and it is defined as an input at this time, the input signal present on the pin itself would be read into the CPU and re-written to the data latch of this particular pin, overwriting the previous content. As long as the pin stays in the input mode, no problem occurs. However, if bit0 is switched into output mode later on, the content of the data latch may now be unknown.

Reading a port reads the values of the port pins. Writing to the port register writes the value to the port latch. When using read-modify-write instructions (BCF, BSF, BTG, etc.) on a port, the value of the port pins is read, the desired operation is performed with this value, and the value is then written to the port latch.

Example 9-5 shows the effect of two sequential read-modify-write instructions on an I/O port

EXAMPLE 9-5: READ MODIFY WRITE INSTRUCTIONS ON AN I/O PORT

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

Note: A pin actively outputting a Low or High should not be driven from external devices in order to change the level on this pin (i.e. “wired-or”, “wired-and”). The resulting high output currents may damage the device.

9.5.2 SUCCESSIVE OPERATIONS ON I/O PORTS

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

FIGURE 9-9: SUCCESSIVE I/O OPERATION

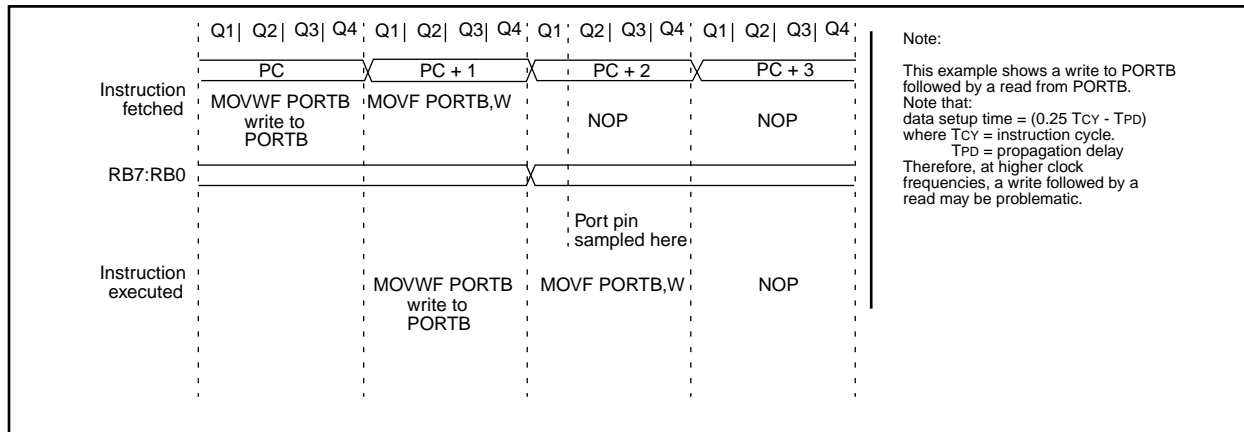


FIGURE 13-5: ASYNCHRONOUS MASTER TRANSMISSION

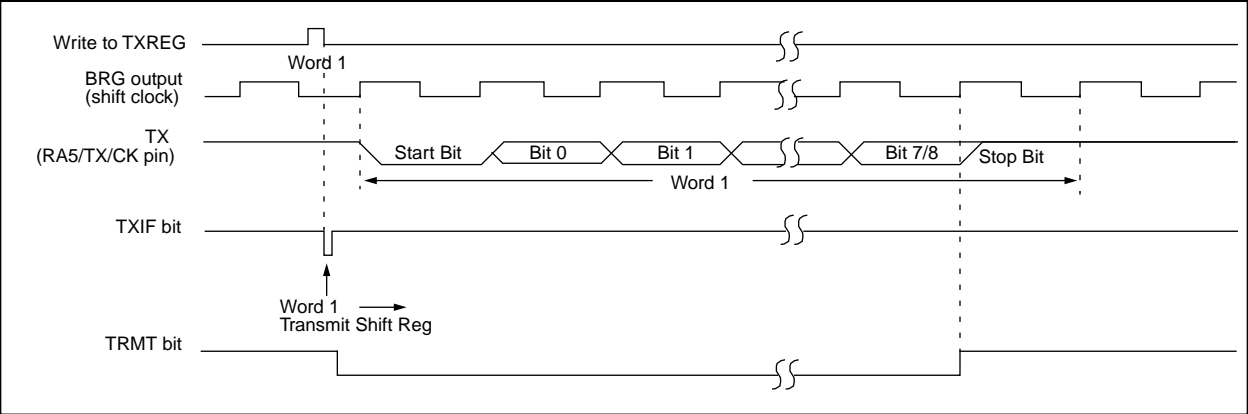


FIGURE 13-6: ASYNCHRONOUS MASTER TRANSMISSION (BACK TO BACK)

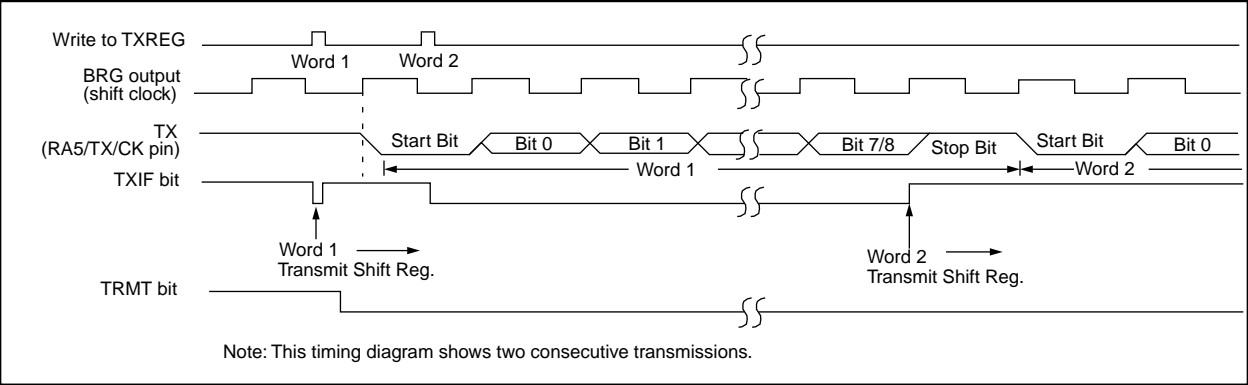


TABLE 13-5: REGISTERS ASSOCIATED WITH ASYNCHRONOUS TRANSMISSION

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other resets (Note1)
16h, Bank 1	PIR	RBIF	TMR3IF	TMR2IF	TMR1IF	CA2IF	CA1IF	TXIF	RCIF	0000 0010	0000 0010
13h, Bank 0	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00u
16h, Bank 0	TXREG	Serial port transmit register								xxxx xxxx	uuuu uuuu
17h, Bank 1	PIE	RBIE	TMR3IE	TMR2IE	TMR1IE	CA2IE	CA1IE	TXIE	RCIE	0000 0000	0000 0000
15h, Bank 0	TXSTA	CSRC	TX9	TXEN	SYNC	—	—	TRMT	TX9D	0000 --1x	0000 --1u
17h, Bank 0	SPBRG	Baud rate generator register								xxxx xxxx	uuuu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented read as a '0', shaded cells are not used for asynchronous transmission.

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

13.3 USART Synchronous Master Mode

In Master Synchronous 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. The synchronous mode is entered by setting the SYNC (TXSTA<4>) bit. In addition, the SPEN (RCSTA<7>) bit is set in order to configure the RA5 and RA4 I/O ports 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 the CSRC (TXSTA<7>) bit.

13.3.1 USART SYNCHRONOUS MASTER TRANSMISSION

The USART transmitter block diagram is shown in Figure 13-3. 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. TXREG is loaded with data in software. The TSR 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 TXREG (if available). Once TXREG transfers the data to the TSR (occurs in one Tcy at the end of the current BRG cycle), TXREG is empty and the TXIF (PIR<1>) bit is set. This interrupt can be enabled/disabled by setting/clearing the TXIE bit (PIE<1>). TXIF will be set regardless of the state of bit TXIE and cannot be cleared in software. It will reset only when new data is loaded into TXREG. While TXIF indicates the status of TXREG, TRMT (TXSTA<1>) shows the status of the TSR. 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 is empty. The TSR is not mapped in data memory, so it is not available to the user.

Transmission is enabled by setting the TXEN (TXSTA<5>) bit. The actual transmission will not occur until TXREG has been loaded with data. The first data bit will be shifted out on the next available rising edge of the clock on the RA5/TX/CK pin. Data out is stable around the falling edge of the synchronous clock (Figure 13-10). The transmission can also be started by first loading TXREG and then setting TXEN. This is advantageous when slow baud rates are selected, since BRG is kept in RESET when the TXEN, CREN, and SREN bits are clear. Setting the TXEN bit will start the BRG, creating a shift clock immediately. Normally when transmission is first started, the TSR is empty, so a transfer to TXREG will result in an immediate transfer to the TSR, resulting in an empty TXREG. Back-to-back transfers are possible.

Clearing TXEN during a transmission will cause the transmission to be aborted and will reset the transmitter. The RA4/RX/DT and RA5/TX/CK pins will revert to hi-impedance. If either CREN or SREN are set during a transmission, the transmission is aborted and the

RA4/RX/DT pin reverts to a hi-impedance state (for a reception). The RA5/TX/CK pin will remain an output if the CSRC bit is set (internal clock). The transmitter logic is not reset, although it is disconnected from the pins. In order to reset the transmitter, the user has to clear the TXEN bit. If the SREN bit is set (to interrupt an ongoing transmission and receive a single word), then after the single word is received, SREN will be cleared and the serial port will revert back to transmitting, since the TXEN bit is still set. The DT line will immediately switch from hi-impedance receive mode to transmit and start driving. To avoid this, 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 TX9D (TXSTA<0>). The ninth bit must be written before writing the 8-bit data to TXREG. This is because a data write to TXREG can result in an immediate transfer of the data to the TSR (if the TSR is empty). If the TSR was empty and TXREG was written before writing the "new" TX9D, the "present" value of TX9D is loaded.

Steps to follow when setting up a Synchronous Master Transmission:

1. Initialize the SPBRG register for the appropriate baud rate (see Baud Rate Generator Section for details).
2. Enable the synchronous master serial port by setting the SYNC, SPEN, and CSRC bits.
3. Ensure that the CREN and SREN bits are clear (these bits override transmission when set).
4. If interrupts are desired, then set the TXIE bit (the GLINTD bit must be clear and the PEIE bit must be set).
5. If 9-bit transmission is desired, then set the TX9 bit.
6. Start transmission by loading data to the TXREG register.
7. If 9-bit transmission is selected, the ninth bit should be loaded in TX9D.
8. Enable the transmission by setting TXEN.

Writing the transmit data to the TXREG, then enabling the transmit (setting TXEN) allows transmission to start sooner than doing these two events in the reverse order.

Note: To terminate a transmission, either clear the SPEN bit, or the TXEN bit. This will reset the transmit logic, so that it will be in the proper state when transmit is re-enabled.

14.4.2 MINIMIZING CURRENT CONSUMPTION

To minimize current consumption, all I/O pins should be either at VDD, or VSS, with no external circuitry drawing current from the I/O pin. I/O pins that are hi-impedance inputs should be pulled high or low externally to avoid switching currents caused by floating inputs. The T0CKI input should be at VDD or VSS. The contributions from on-chip pull-ups on PORTB should also be considered, and disabled when possible.

14.5 Code Protection

The code in the program memory can be protected by selecting the microcontroller in code protected mode (PM2:PM0 = '000').

Note: PM2 does not exist on the PIC17C42. To select code protected microcontroller mode, PM1:PM0 = '00'.

In this mode, instructions that are in the on-chip program memory space, can continue to read or write the program memory. An instruction that is executed outside of the internal program memory range will be inhibited from writing to or reading from program memory.

Note: Microchip does not recommend code protecting windowed devices.

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

15.2 Q Cycle Activity

Each instruction cycle (Tcy) is comprised of four Q cycles (Q1-Q4). The Q cycles provide the timing/designation for the Decode, Read, Execute, Write etc., of each instruction cycle. The following diagram shows the relationship of the Q cycles to the instruction cycle.

The 4 Q cycles that make up an instruction cycle (Tcy) can be generalized as:

Q1: Instruction Decode Cycle or forced NOP

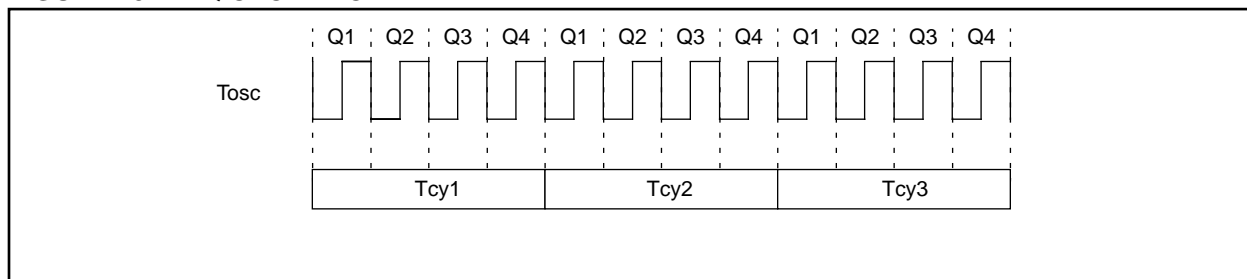
Q2: Instruction Read Cycle or NOP

Q3: Instruction Execute

Q4: Instruction Write Cycle or NOP

Each instruction will show the detailed Q cycle operation for the instruction.

FIGURE 15-2: Q CYCLE ACTIVITY



ADDLW

ADD Literal to WREG

Syntax: [*label*] ADDLW k

Operands: $0 \leq k \leq 255$

Operation: (WREG) + k → (WREG)

Status Affected: OV, C, DC, Z

Encoding:

1011	0001	kkkk	kkkk
------	------	------	------

Description: The contents of WREG are added to the 8-bit literal 'k' and the result is placed in WREG.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Execute	Write to WREG

Example: ADDLW 0x15

Before Instruction
WREG = 0x10

After Instruction
WREG = 0x25

ADDWF

ADD WREG to f

Syntax: [*label*] ADDWF f,d

Operands: $0 \leq f \leq 255$
 $d \in [0,1]$

Operation: (WREG) + (f) → (dest)

Status Affected: OV, C, DC, Z

Encoding:

0000	111d	ffff	ffff
------	------	------	------

Description: Add WREG to register 'f'. If 'd' is 0 the result is stored in WREG. If 'd' is 1 the result is stored back in register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Execute	Write to destination

Example: ADDWF REG, 0

Before Instruction
WREG = 0x17
REG = 0xC2

After Instruction
WREG = 0xD9
REG = 0xC2

CALL

Subroutine Call

Syntax: [label] CALL k

Operands: 0 ≤ k ≤ 4095

Operation: PC+ 1→ TOS, k → PC<12:0>, k<12:8> → PCLATH<4:0>; PC<15:13> → PCLATH<7:5>

Status Affected: None

Encoding:

111k	kkkk	kkkk	kkkk
------	------	------	------

Description: Subroutine call within 8K page. First, return address (PC+1) is pushed onto the stack. The 13-bit value is loaded into PC bits<12:0>. Then the upper-eight bits of the PC are copied into PCLATH. Call is a two-cycle instruction. See LCALL for calls outside 8K memory space.

Words: 1
Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'<7:0>	Execute	NOP
Forced NOP	NOP	Execute	NOP

Example: HERE CALL THERE

Before Instruction
PC = Address (HERE)

After Instruction
PC = Address (THERE)
TOS = Address (HERE + 1)

CLRF

Clear f

Syntax: [label] CLRF f,s

Operands: 0 ≤ f ≤ 255

Operation: 00h → f, s ∈ [0,1]
00h → dest

Status Affected: None

Encoding:

0010	100s	ffff	ffff
------	------	------	------

Description: Clears the contents of the specified register(s).
s = 0: Data memory location 'f' and WREG are cleared.
s = 1: Data memory location 'f' is cleared.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Execute	Write register 'f' and other specified register

Example: CLRF FLAG_REG

Before Instruction
FLAG_REG = 0x5A

After Instruction
FLAG_REG = 0x00

IORWF		Inclusive OR WREG with f						
Syntax:	[<i>label</i>] IORWF f,d							
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$							
Operation:	(WREG) .OR. (f) \rightarrow (dest)							
Status Affected:	Z							
Encoding:	<table><tr><td>0000</td><td>100d</td><td>ffff</td><td>ffff</td></tr></table>				0000	100d	ffff	ffff
0000	100d	ffff	ffff					
Description:	Inclusive OR WREG with register 'f'. If 'd' is 0 the result is placed in WREG. If 'd' is 1 the result is placed back in register 'f'.							
Words:	1							
Cycles:	1							
Q Cycle Activity:								
	Q1	Q2	Q3	Q4				
	Decode	Read register 'f'	Execute	Write to destination				

Example: IORWF RESULT, 0

Before Instruction

RESULT = 0x13
WREG = 0x91

After Instruction

RESULT = 0x13
WREG = 0x93

LCALL	Long Call												
Syntax:	[<i>label</i>] LCALL k												
Operands:	$0 \leq k \leq 255$												
Operation:	PC + 1 \rightarrow TOS; k \rightarrow PCL, (PCLATH) \rightarrow PCH												
Status Affected:	None												
Encoding:	<table><tr><td>1011</td><td>0111</td><td>kkkk</td><td>kkkk</td></tr></table>	1011	0111	kkkk	kkkk								
1011	0111	kkkk	kkkk										
Description:	<p>LCALL allows an unconditional subroutine call to anywhere within the 64k program memory space.</p> <p>First, the return address (PC + 1) is pushed onto the stack. A 16-bit destination address is then loaded into the program counter. The lower 8-bits of the destination address is embedded in the instruction. The upper 8-bits of PC is loaded from PC high holding latch, PCLATH.</p>												
Words:	1												
Cycles:	2												
Q Cycle Activity:													
	<table><tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr><tr><td>Decode</td><td>Read literal 'k'</td><td>Execute</td><td>Write register PCL</td></tr><tr><td>Forced NOP</td><td>NOP</td><td>Execute</td><td>NOP</td></tr></table>	Q1	Q2	Q3	Q4	Decode	Read literal 'k'	Execute	Write register PCL	Forced NOP	NOP	Execute	NOP
Q1	Q2	Q3	Q4										
Decode	Read literal 'k'	Execute	Write register PCL										
Forced NOP	NOP	Execute	NOP										

Example: MOVLW HIGH(SUBROUTINE)
MOVWF WREG, PCLATH
LCALL LOW(SUBROUTINE)

Before Instruction

SUBROUTINE = 16-bit Address
PC = ?

After Instruction

PC = Address (SUBROUTINE)

PIC17C4X

Applicable Devices 42 R42 42A 43 R43 44

TABLE 17-1: CROSS REFERENCE OF DEVICE SPECS FOR OSCILLATOR CONFIGURATIONS AND FREQUENCIES OF OPERATION (COMMERCIAL DEVICES)

OSC	PIC17C42-16	PIC17C42-25
RC	VDD: 4.5V to 5.5V IDD: 6 mA max. IPD: 5 μ A max. at 5.5V (WDT disabled) Freq: 4 MHz max.	VDD: 4.5V to 5.5V IDD: 6 mA max. IPD: 5 μ A max. at 5.5V (WDT disabled) Freq: 4 MHz max.
XT	VDD: 4.5V to 5.5V IDD: 24 mA max. IPD: 5 μ A max. at 5.5V (WDT disabled) Freq: 16 MHz max.	VDD: 4.5V to 5.5V IDD: 38 mA max. IPD: 5 μ A max. at 5.5V (WDT disabled) Freq: 25 MHz max.
EC	VDD: 4.5V to 5.5V IDD: 24 mA max. IPD: 5 μ A max. at 5.5V (WDT disabled) Freq: 16 MHz max.	VDD: 4.5V to 5.5V IDD: 38 mA max. IPD: 5 μ A max. at 5.5V (WDT disabled) Freq: 25 MHz max.
LF	VDD: 4.5V to 5.5V IDD: 150 μ A max. at 32 kHz (WDT enabled) IPD: 5 μ A max. at 5.5V (WDT disabled) Freq: 2 MHz max.	VDD: 4.5V to 5.5V IDD: 150 μ A max. at 32 kHz (WDT enabled) IPD: 5 μ A max. at 5.5V (WDT disabled) Freq: 2 MHz max.

FIGURE 17-7: CAPTURE TIMINGS

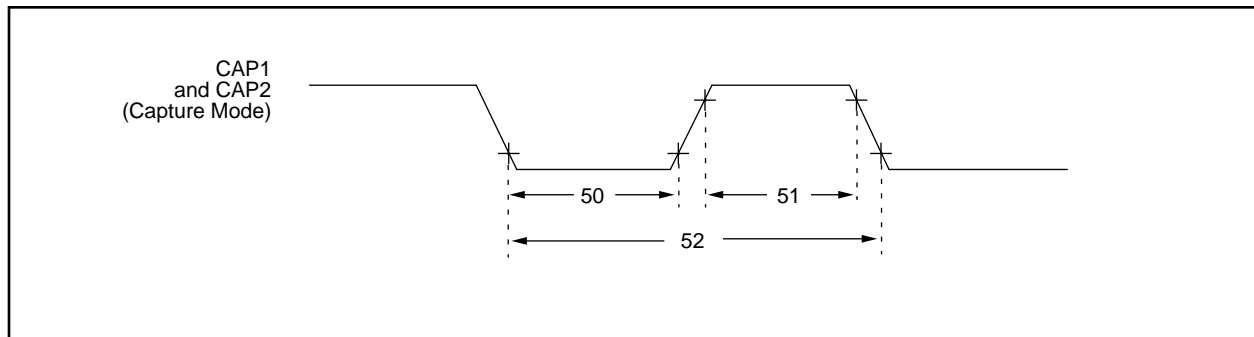


TABLE 17-7: CAPTURE REQUIREMENTS

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
50	TccL	Capture1 and Capture2 input low time	10 *	—	—	ns	
51	TccH	Capture1 and Capture2 input high time	10 *	—	—	ns	
52	TccP	Capture1 and Capture2 input period	$\frac{2 T_{CY}}{N}$ §	—	—	ns	N = prescale value (4 or 16)

* 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.

§ This specification ensured by design.

FIGURE 17-8: PWM TIMINGS

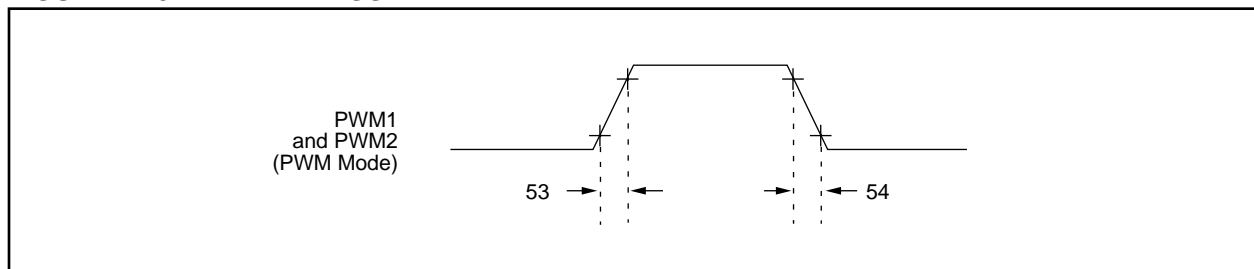


TABLE 17-8: PWM REQUIREMENTS

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
53	TccR	PWM1 and PWM2 output rise time	—	10 *	35 *§	ns	
54	TccF	PWM1 and PWM2 output fall time	—	10 *	35 *§	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.

§ This specification ensured by design.

FIGURE 19-3: CLKOUT AND I/O TIMING

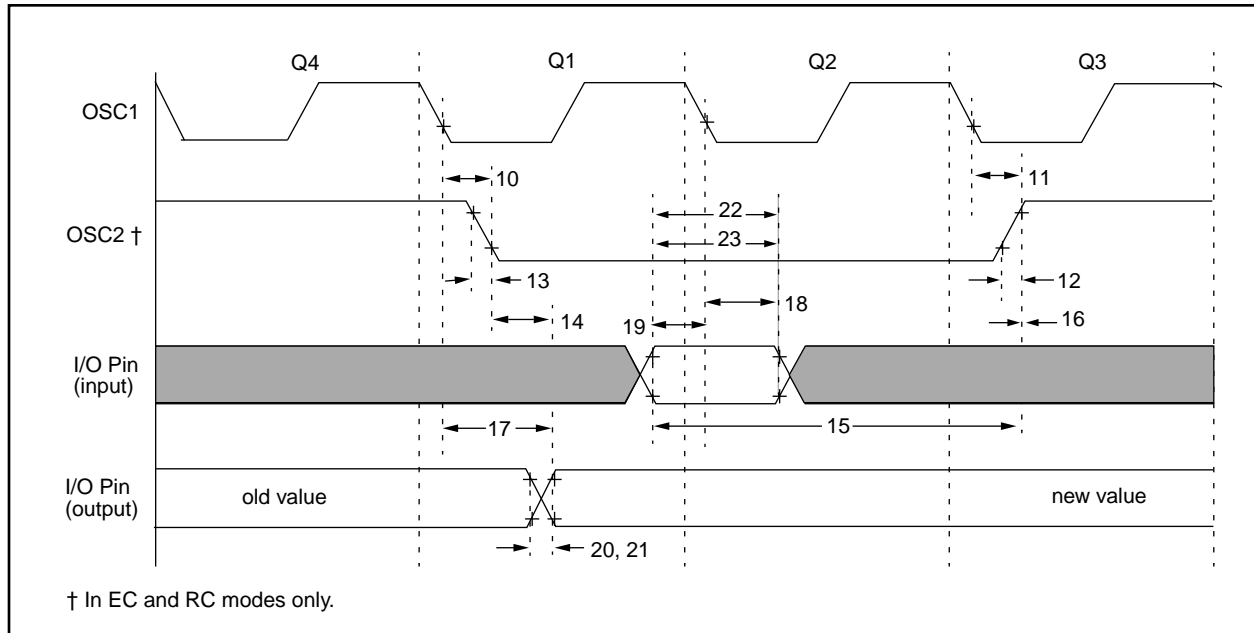


TABLE 19-3: CLKOUT AND I/O TIMING REQUIREMENTS

Parameter No.	Sym	Characteristic		Min	Typ†	Max	Units	Conditions
10	TosH2ckL	OSC1↓ to CLKOUT↓		—	15 ‡	30 ‡	ns	Note 1
11	TosH2ckH	OSC1↓ to CLKOUT↑		—	15 ‡	30 ‡	ns	Note 1
12	TckR	CLKOUT rise time		—	5 ‡	15 ‡	ns	Note 1
13	TckF	CLKOUT fall time		—	5 ‡	15 ‡	ns	Note 1
14	TckH2ioV	CLKOUT ↑ to Port out valid	PIC17CR42/42A/43/R43/44	—	—	0.5TCY + 20 ‡	ns	Note 1
			PIC17LCR42/42A/43/R43/44	—	—	0.5TCY + 50 ‡	ns	Note 1
15	TioV2ckH	Port in valid before CLKOUT↑	PIC17CR42/42A/43/R43/44	0.25TCY + 25 ‡	—	—	ns	Note 1
			PIC17LCR42/42A/43/R43/44	0.25TCY + 50 ‡	—	—	ns	Note 1
16	TckH2iol	Port in hold after CLKOUT↑		0 ‡	—	—	ns	Note 1
17	TosH2ioV	OSC1↓ (Q1 cycle) to Port out valid		—	—	100 ‡	ns	
18	TosH2iol	OSC1↓ (Q2 cycle) to Port input invalid (I/O in hold time)		0 ‡	—	—	ns	
19	TioV2osH	Port input valid to OSC1↓ (I/O in setup time)		30 ‡	—	—	ns	
20	TioR	Port output rise time		—	10 ‡	35 ‡	ns	
21	TioF	Port output fall time		—	10 ‡	35 ‡	ns	
22	TinHL	INT pin high or low time		25 *	—	—	ns	
23	TrbHL	RB7:RB0 change INT high or low time		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.

‡ These parameters are for design guidance only and are not tested, nor characterized.

Note 1: Measurements are taken in EC Mode where CLKOUT output is 4 x T_{osc}.

FIGURE 20-7: TYPICAL I_{DD} vs. FREQUENCY (EXTERNAL CLOCK 25°C)

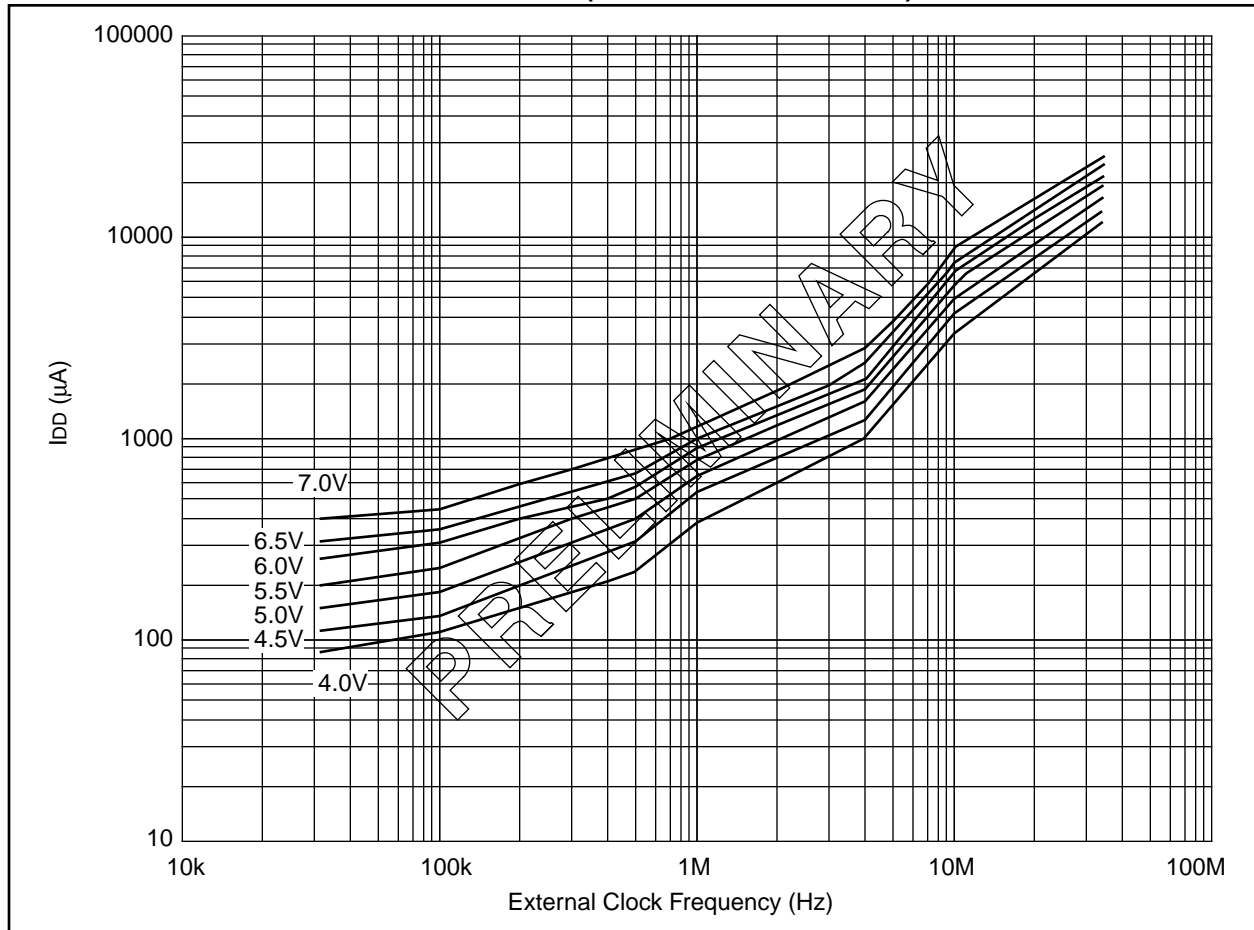
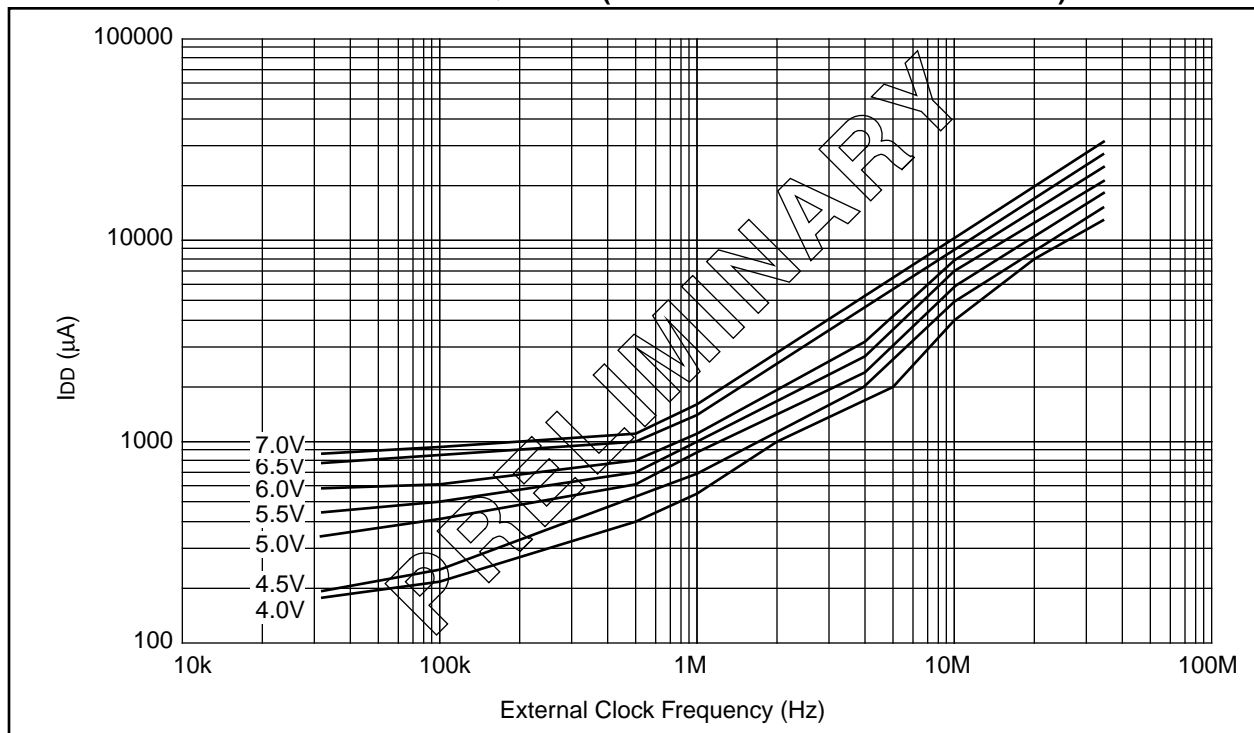


FIGURE 20-8: MAXIMUM I_{DD} vs. FREQUENCY (EXTERNAL CLOCK 125°C TO -40°C)



PIC17C4X

Applicable Devices 42 R42 42A 43 R43 44

FIGURE 20-9: TYPICAL I_{PD} vs. V_{DD} WATCHDOG DISABLED 25°C

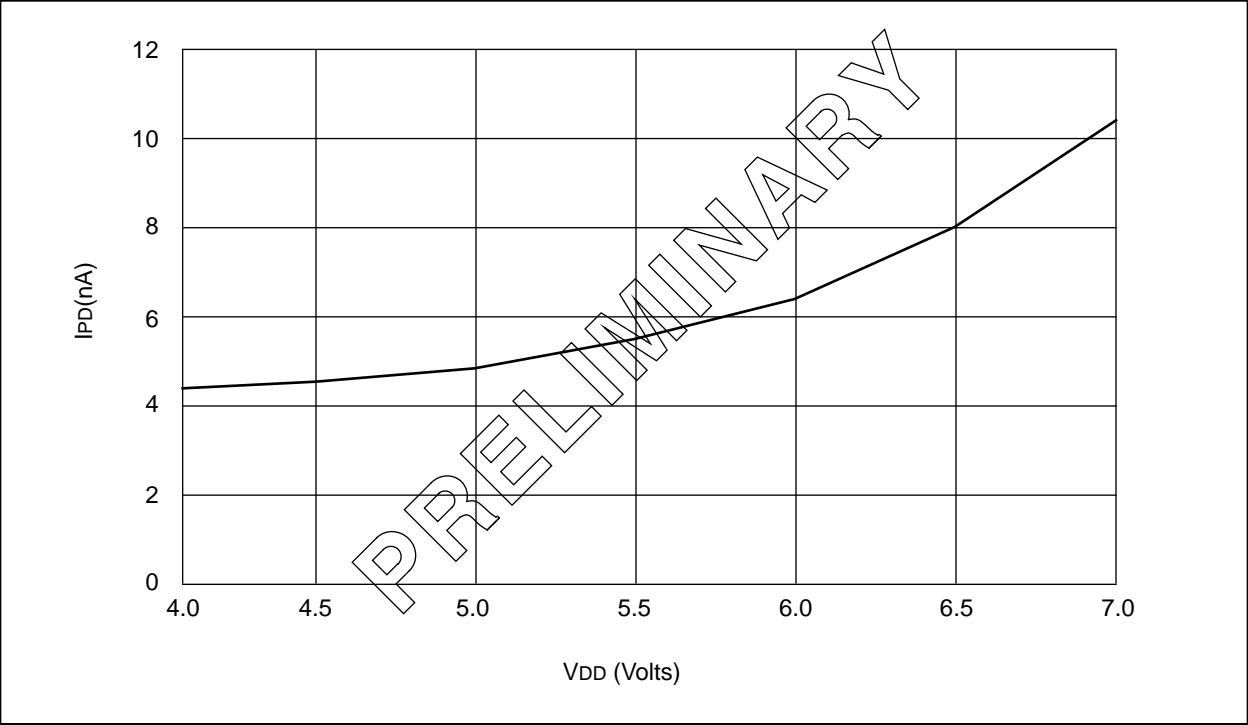
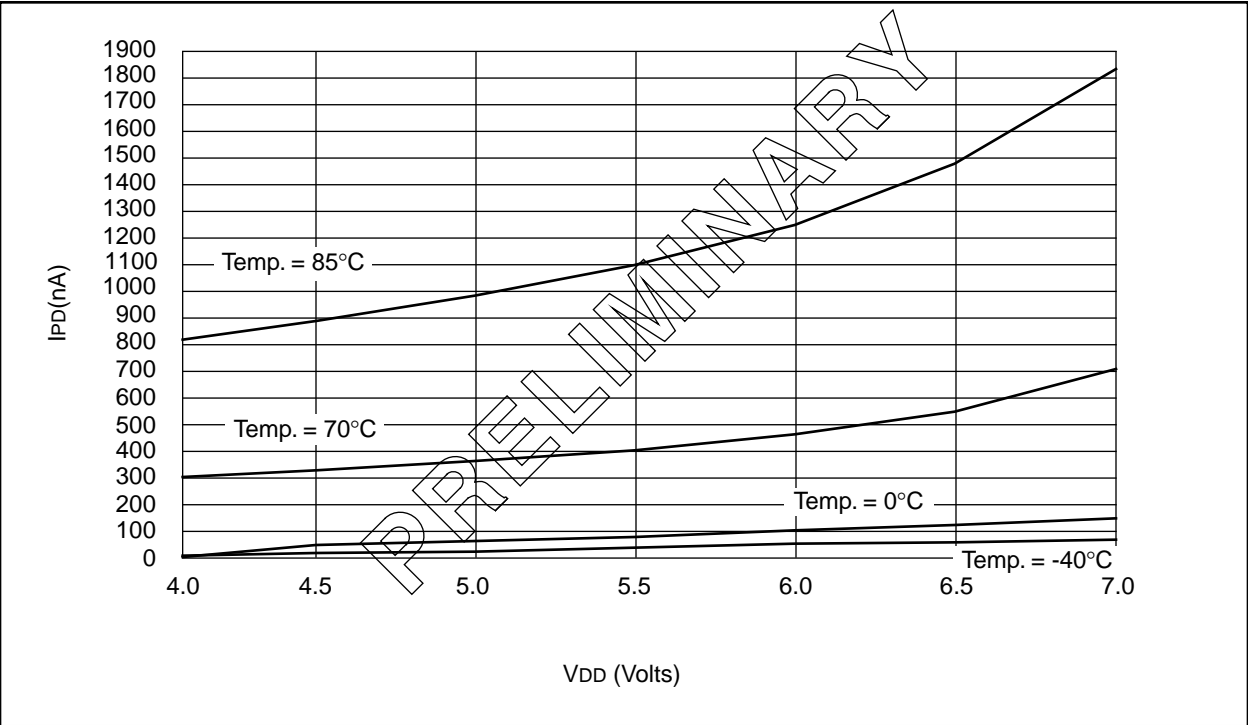


FIGURE 20-10: MAXIMUM I_{PD} vs. V_{DD} WATCHDOG DISABLED



PIC17C4X

Applicable Devices 42 R42 42A 43 R43 44

FIGURE 20-17: I_{OH} vs. V_{OL} , $V_{DD} = 5V$

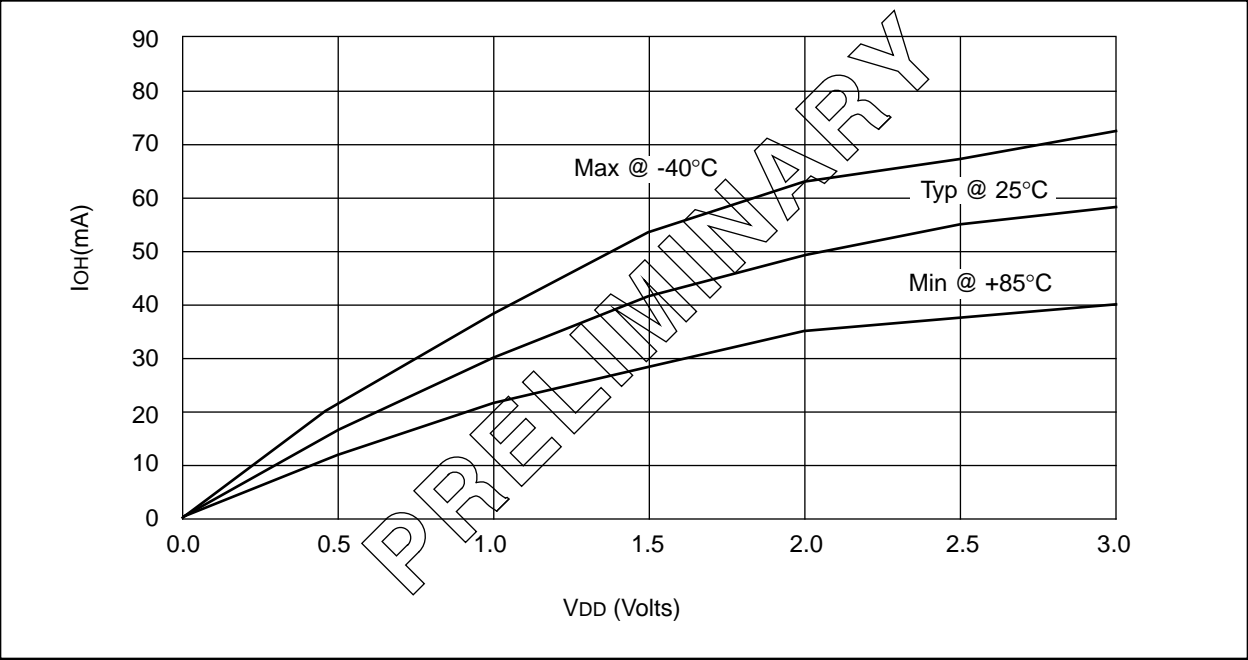


FIGURE 20-18: V_{TH} (INPUT THRESHOLD VOLTAGE) OF I/O PINS (TTL) vs. V_{DD}

