



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	Active
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	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/pic17c44-25e-p

7.1 Table Writes to Internal Memory

A table write operation to internal memory causes a long write operation. The long write is necessary for programming the internal EPROM. Instruction execution is halted while in a long write cycle. The long write will be terminated by any enabled interrupt. To ensure that the EPROM location has been well programmed, a minimum programming time is required (see specification #D114). Having only one interrupt enabled to terminate the long write ensures that no unintentional interrupts will prematurely terminate the long write.

The sequence of events for programming an internal program memory location should be:

1. Disable all interrupt sources, except the source to terminate EPROM program write.
2. Raise $\overline{\text{MCLR}}$ /VPP pin to the programming voltage.
3. Clear the WDT.
4. Do the table write. The interrupt will terminate the long write.
5. Verify the memory location (table read).

Note: Programming requirements must be met. See timing specification in electrical specifications for the desired device. Violating these specifications (including temperature) may result in EPROM locations that are not fully programmed and may lose their state over time.

7.1.1 TERMINATING LONG WRITES

An interrupt source or reset are the only events that terminate a long write operation. Terminating the long write from an interrupt source requires that the interrupt enable and flag bits are set. The GLINTD bit only enables the vectoring to the interrupt address.

If the T0CKI, RA0/INT, or TMR0 interrupt source is used to terminate the long write; the interrupt flag, of the highest priority enabled interrupt, will terminate the long write and automatically be cleared.

Note 1: If an interrupt is pending, the TABLWT is aborted (an NOP is executed). The highest priority pending interrupt, from the T0CKI, RA0/INT, or TMR0 sources that is enabled, has its flag cleared.

Note 2: If the interrupt is not being used for the program write timing, the interrupt should be disabled. This will ensure that the interrupt is not lost, nor will it terminate the long write prematurely.

If a peripheral interrupt source is used to terminate the long write, the interrupt enable and flag bits must be set. The interrupt flag will not be automatically cleared upon the vectoring to the interrupt vector address.

If the GLINTD bit is cleared prior to the long write, when the long write is terminated, the program will branch to the interrupt vector.

If the GLINTD bit is set prior to the long write, when the long write is terminated, the program will not vector to the interrupt address.

TABLE 7-1: INTERRUPT - TABLE WRITE INTERACTION

Interrupt Source	GLINTD	Enable Bit	Flag Bit	Action
RA0/INT, TMR0, T0CKI	0	1	1	Terminate long table write (to internal program memory), branch to interrupt vector (branch clears flag bit).
	0	1	0	None
	1	0	x	None
	1	1	1	Terminate table write, do not branch to interrupt vector (flag is automatically cleared).
Peripheral	0	1	1	Terminate table write, branch to interrupt vector.
	0	1	0	None
	1	0	x	None
	1	1	1	Terminate table write, do not branch to interrupt vector (flag is set).

11.1 Timer0 Operation

When the T0CS (T0STA<5>) bit is set, TMR0 increments on the internal clock. When T0CS is clear, TMR0 increments on the external clock (RA1/T0CKI pin). The external clock edge can be configured in software. When the T0SE (T0STA<6>) bit is set, the timer will increment on the rising edge of the RA1/T0CKI pin. When T0SE is clear, the timer will increment on the falling edge of the RA1/T0CKI pin. The prescaler can be programmed to introduce a prescale of 1:1 to 1:256. The timer increments from 0000h to FFFFh and rolls over to 0000h. On overflow, the TMR0 Interrupt Flag bit (T0IF) is set. The TMR0 interrupt can be masked by clearing the corresponding TMR0 Interrupt Enable bit (T0IE). The TMR0 Interrupt Flag bit (T0IF) is automatically cleared when vectoring to the TMR0 interrupt vector.

11.2 Using Timer0 with External Clock

When the external clock input is used for Timer0, it is synchronized with the internal phase clocks. Figure 11-3 shows the synchronization of the external clock. This synchronization is done after the prescaler. The output of the prescaler (PSOUT) is sampled twice in every instruction cycle to detect a rising or a falling edge. The timing requirements for the external clock are detailed in the electrical specification section for the desired device.

11.2.1 DELAY FROM EXTERNAL CLOCK EDGE

Since the prescaler output is synchronized with the internal clocks, there is a small delay from the time the external clock edge occurs to the time TMR0 is actually incremented. Figure 11-3 shows that this delay is between 3TOSC and 7TOSC. Thus, for example, measuring the interval between two edges (e.g. period) will be accurate within $\pm 4TOSC$ (± 121 ns @ 33 MHz).

FIGURE 11-2: TIMER0 MODULE BLOCK DIAGRAM

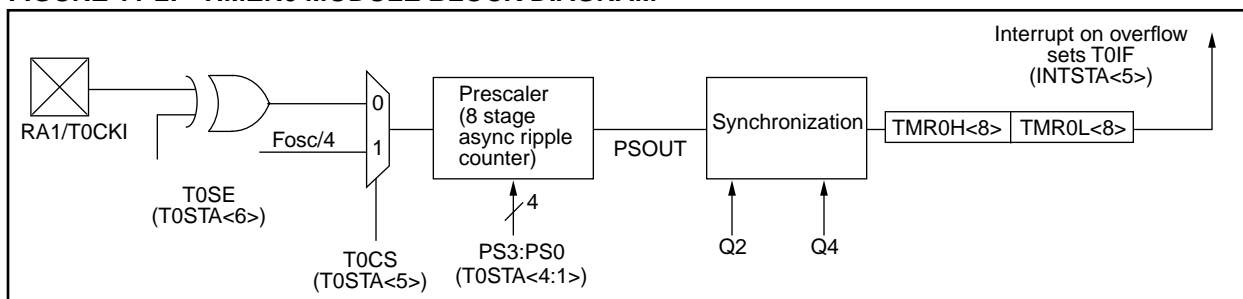
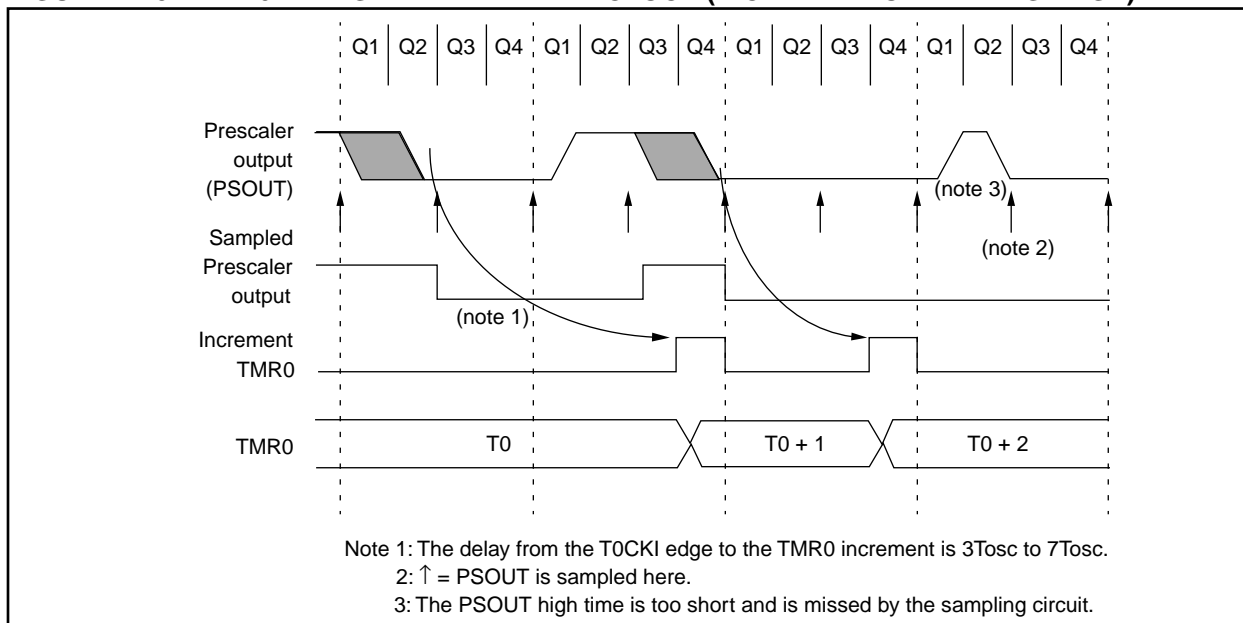


FIGURE 11-3: TMR0 TIMING WITH EXTERNAL CLOCK (INCREMENT ON FALLING EDGE)



11.3 Read/Write Consideration for TMR0

Although TMR0 is a 16-bit timer/counter, only 8-bits at a time can be read or written during a single instruction cycle. Care must be taken during any read or write.

11.3.1 READING 16-BIT VALUE

The problem in reading the entire 16-bit value is that after reading the low (or high) byte, its value may change from FFh to 00h.

Example 11-1 shows a 16-bit read. To ensure a proper read, interrupts must be disabled during this routine.

EXAMPLE 11-1: 16-BIT READ

```

MOVFP  TMR0L, TMPLO    ;read low tmr0
MOVFP  TMR0H, TMPHI    ;read high tmr0
MOVFP  TMPLO, WREG      ;tmplo -> wreg
CPFSLT TMR0L           ;tmr0l < wreg?
RETURN                ;no then return
MOVFP  TMR0L, TMPLO    ;read low tmr0
MOVFP  TMR0H, TMPHI    ;read high tmr0
RETURN                ;return
    
```

11.3.2 WRITING A 16-BIT VALUE TO TMR0

Since writing to either TMR0L or TMR0H will effectively inhibit increment of that half of the TMR0 in the next cycle (following write), but not inhibit increment of the other half, the user must write to TMR0L first and TMR0H next in two consecutive instructions, as shown in Example 11-2. The interrupt must be disabled. Any write to either TMR0L or TMR0H clears the prescaler.

EXAMPLE 11-2: 16-BIT WRITE

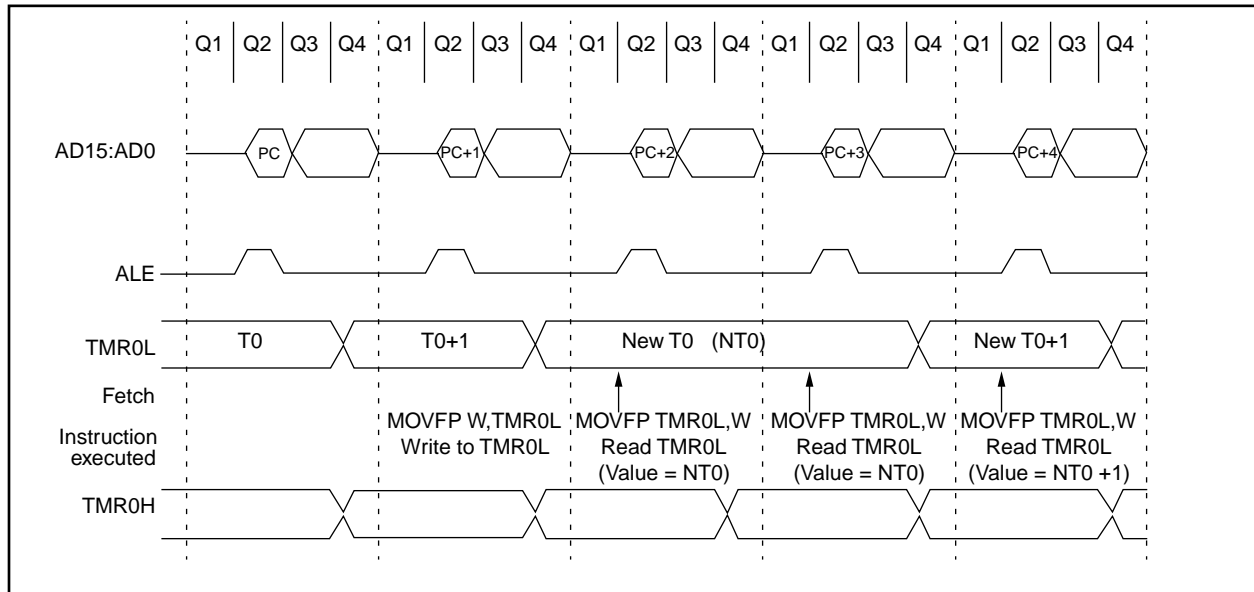
```

BSF    CPUSTA, GLINTD ; Disable interrupt
MOVFP  RAM_L, TMR0L   ;
MOVFP  RAM_H, TMR0H   ;
BCF    CPUSTA, GLINTD ; Done, enable interrupt
    
```

11.4 Prescaler Assignments

Timer0 has an 8-bit prescaler. The prescaler assignment is fully under software control; i.e., it can be changed “on the fly” during program execution. When changing the prescaler assignment, clearing the prescaler is recommended before changing assignment. The value of the prescaler is “unknown,” and assigning a value that is less than the present value makes it difficult to take this unknown time into account.

FIGURE 11-4: TMR0 TIMING: WRITE HIGH OR LOW BYTE



12.1.3.1 PWM PERIODS

The period of the PWM1 output is determined by Timer1 and its period register (PR1). The period of the PWM2 output can be software configured to use either Timer1 or Timer2 as the time-base. When TM2PW2 bit (PW2DCL<5>) is clear, the time-base is determined by TMR1 and PR1. When TM2PW2 is set, the time-base is determined by Timer2 and PR2.

Running two different PWM outputs on two different timers allows different PWM periods. Running both PWMs from Timer1 allows the best use of resources by freeing Timer2 to operate as an 8-bit timer. Timer1 and Timer2 can not be used as a 16-bit timer if either PWM is being used.

The PWM periods can be calculated as follows:

$$\begin{aligned} \text{period of PWM1} &= [(PR1) + 1] \times 4T_{osc} \\ \text{period of PWM2} &= [(PR1) + 1] \times 4T_{osc} \quad \text{or} \\ &[(PR2) + 1] \times 4T_{osc} \end{aligned}$$

The duty cycle of PWMx is determined by the 10-bit value DCx<9:0>. The upper 8-bits are from register PWxDCH and the lower 2-bits are from PWxDCL<7:6> (PWxDCH:PWxDCL<7:6>). Table 12-3 shows the maximum PWM frequency (FPWM) given the value in the period register.

The number of bits of resolution that the PWM can achieve depends on the operation frequency of the device as well as the PWM frequency (FPWM).

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

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

The PWMx duty cycle is as follows:

$$\text{PWMx Duty Cycle} = (DCx) \times T_{osc}$$

where DCx represents the 10-bit value from PWxDCH:PWxDCL.

If DCx = 0, then the duty cycle is zero. If PRx = PWxDCH, then the PWM output will be low for one to four Q-clock (depending on the state of the PWxDCL<7:6> bits). For a Duty Cycle to be 100%, the PWxDCH value must be greater than the PRx value.

The duty cycle registers for both PWM outputs are double buffered. When the user writes to these registers, they are stored in master latches. When TMR1 (or TMR2) overflows and a new PWM period begins, the master latch values are transferred to the slave latches and the PWMx pin is forced high.

Note: For PW1DCH, PW1DCL, PW2DCH and PW2DCL registers, a write operation writes to the "master latches" while a read operation reads the "slave latches". As a result, the user may not read back what was just written to the duty cycle registers.

The user should also avoid any "read-modify-write" operations on the duty cycle registers, such as: ADDWF PW1DCH. This may cause duty cycle outputs that are unpredictable.

TABLE 12-3: PWM FREQUENCY vs. RESOLUTION AT 25 MHz

PWM Frequency	Frequency (kHz)				
	24.4	48.8	65.104	97.66	390.6
PRx Value	0xFF	0x7F	0x5F	0x3F	0x0F
High Resolution	10-bit	9-bit	8.5-bit	8-bit	6-bit
Standard Resolution	8-bit	7-bit	6.5-bit	6-bit	4-bit

12.1.3.2 PWM INTERRUPTS

The PWM module makes use of TMR1 or TMR2 interrupts. A timer interrupt is generated when TMR1 or TMR2 equals its period register and is cleared to zero. This interrupt also marks the beginning of a PWM cycle. The user can write new duty cycle values before the timer roll-over. The TMR1 interrupt is latched into the TMR1IF bit and the TMR2 interrupt is latched into the TMR2IF bit. These flags must be cleared in software.

12.1.3.3 EXTERNAL CLOCK SOURCE

The PWMs will operate regardless of the clock source of the timer. The use of an external clock has ramifications that must be understood. Because the external TCLK12 input is synchronized internally (sampled once per instruction cycle), the time TCLK12 changes to the time the timer increments will vary by as much as TCY (one instruction cycle). This will cause jitter in the duty cycle as well as the period of the PWM output.

This jitter will be $\pm TCY$, unless the external clock is synchronized with the processor clock. Use of one of the PWM outputs as the clock source to the TCLKx input, will supply a synchronized clock.

In general, when using an external clock source for PWM, its frequency should be much less than the device frequency (Fosc).

TABLE 13-4: BAUD RATES FOR ASYNCHRONOUS MODE

BAUD RATE (K)	FOSC = 33 MHz			FOSC = 25 MHz			FOSC = 20 MHz			FOSC = 16 MHz		
	KBAUD	%ERROR	SPBRG value (decimal)	KBAUD	%ERROR	SPBRG value (decimal)	KBAUD	%ERROR	SPBRG value (decimal)	KBAUD	%ERROR	SPBRG value (decimal)
0.3	NA	—	—	NA	—	—	NA	—	—	NA	—	—
1.2	NA	—	—	NA	—	—	1.221	+1.73	255	1.202	+0.16	207
2.4	2.398	-0.07	214	2.396	0.14	162	2.404	+0.16	129	2.404	+0.16	103
9.6	9.548	-0.54	53	9.53	-0.76	40	9.469	-1.36	32	9.615	+0.16	25
19.2	19.09	-0.54	26	19.53	+1.73	19	19.53	+1.73	15	19.23	+0.16	12
76.8	73.66	-4.09	6	78.13	+1.73	4	78.13	+1.73	3	83.33	+8.51	2
96	103.12	+7.42	4	97.65	+1.73	3	104.2	+8.51	2	NA	—	—
300	257.81	-14.06	1	390.63	+30.21	0	312.5	+4.17	0	NA	—	—
500	515.62	+3.13	0	NA	—	—	NA	—	—	NA	—	—
HIGH	515.62	—	0	—	—	0	312.5	—	0	250	—	0
LOW	2.014	—	255	1.53	—	255	1.221	—	255	0.977	—	255

BAUD RATE (K)	FOSC = 10 MHz			FOSC = 7.159 MHz			FOSC = 5.068 MHz		
	KBAUD	%ERROR	SPBRG value (decimal)	KBAUD	%ERROR	SPBRG value (decimal)	KBAUD	%ERROR	SPBRG value (decimal)
0.3	NA	—	—	NA	—	—	0.31	+3.13	255
1.2	1.202	+0.16	129	1.203	-0.23	92	1.2	0	65
2.4	2.404	+0.16	64	2.380	-0.83	46	2.4	0	32
9.6	9.766	+1.73	15	9.322	-2.90	11	9.9	-3.13	7
19.2	19.53	+1.73	7	18.64	-2.90	5	19.8	+3.13	3
76.8	78.13	+1.73	1	NA	—	—	79.2	+3.13	0
96	NA	—	—	NA	—	—	NA	—	—
300	NA	—	—	NA	—	—	NA	—	—
500	NA	—	—	NA	—	—	NA	—	—
HIGH	156.3	—	0	111.9	—	0	79.2	—	0
LOW	0.610	—	255	0.437	—	255	0.309	—	255

BAUD RATE (K)	FOSC = 3.579 MHz			FOSC = 1 MHz			FOSC = 32.768 kHz		
	KBAUD	%ERROR	SPBRG value (decimal)	KBAUD	%ERROR	SPBRG value (decimal)	KBAUD	%ERROR	SPBRG value (decimal)
0.3	0.301	+0.23	185	0.300	+0.16	51	0.256	-14.67	1
1.2	1.190	-0.83	46	1.202	+0.16	12	NA	—	—
2.4	2.432	+1.32	22	2.232	-6.99	6	NA	—	—
9.6	9.322	-2.90	5	NA	—	—	NA	—	—
19.2	18.64	-2.90	2	NA	—	—	NA	—	—
76.8	NA	—	—	NA	—	—	NA	—	—
96	NA	—	—	NA	—	—	NA	—	—
300	NA	—	—	NA	—	—	NA	—	—
500	NA	—	—	NA	—	—	NA	—	—
HIGH	55.93	—	0	15.63	—	0	0.512	—	0
LOW	0.218	—	255	0.061	—	255	0.002	—	255

FIGURE 13-5: ASYNCHRONOUS MASTER TRANSMISSION

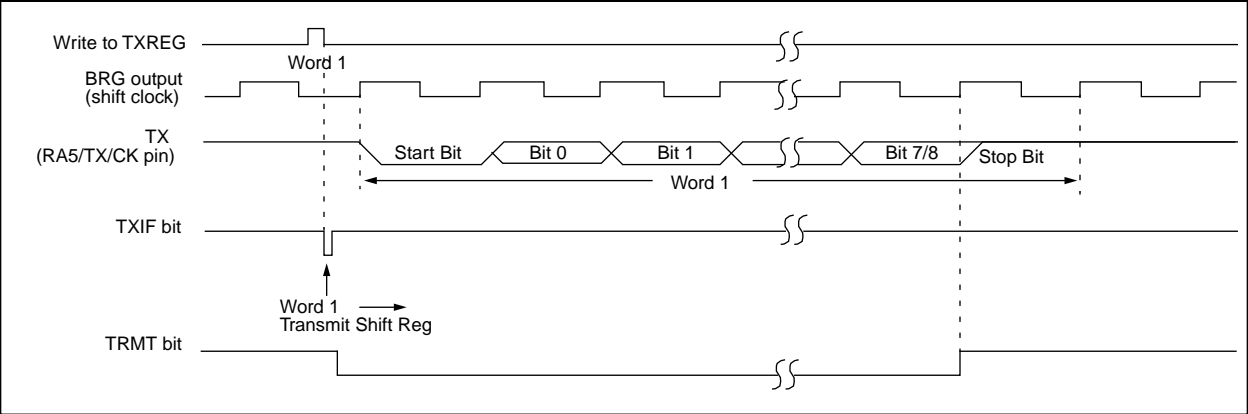


FIGURE 13-6: ASYNCHRONOUS MASTER TRANSMISSION (BACK TO BACK)

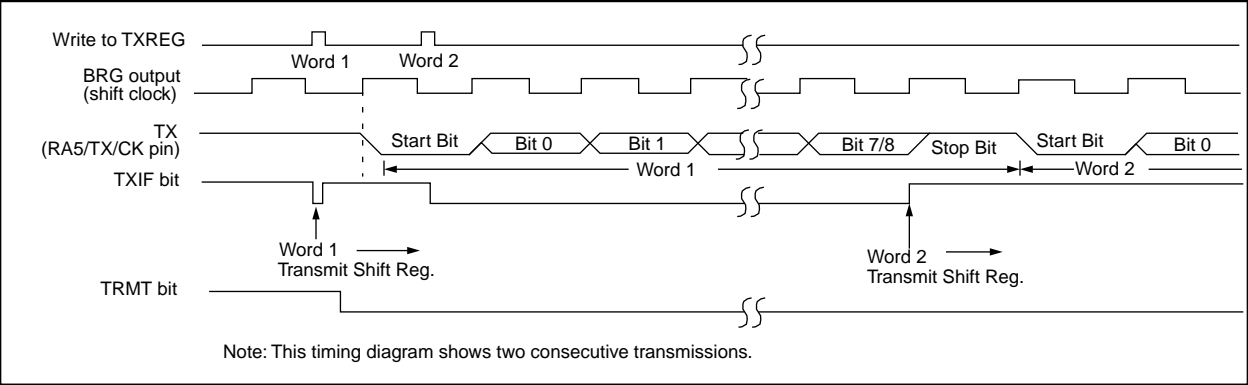


TABLE 13-5: REGISTERS ASSOCIATED WITH ASYNCHRONOUS TRANSMISSION

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.

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.

PIC17C4X

DCFSNZ Decrement f, skip if not 0

Syntax: `[label] DCFSNZ f,d`

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

Operation: $(f) - 1 \rightarrow (\text{dest});$
 skip if not 0

Status Affected: None

Encoding:

0010	011d	ffff	ffff
------	------	------	------

Description: The contents of register 'f' are decremented. If 'd' is 0 the result is placed in WREG. If 'd' is 1 the result is placed back in register 'f'.
 If the result is not 0, the next instruction, which is already fetched, is discarded, and an NOP is executed instead making it a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

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

If skip:

Q1	Q2	Q3	Q4
Forced NOP	NOP	Execute	NOP

Example: HERE DCFSNZ TEMP, 1
 ZERO :
 NZERO :

Before Instruction

TEMP_VALUE = ?

After Instruction

TEMP_VALUE = TEMP_VALUE - 1,
 If TEMP_VALUE = 0;
 PC = Address (ZERO)
 If TEMP_VALUE \neq 0;
 PC = Address (NZERO)

GOTO Unconditional Branch

Syntax: `[label] GOTO k`

Operands: $0 \leq k \leq 8191$

Operation: $k \rightarrow PC<12:0>;$
 $k<12:8> \rightarrow PCLATH<4:0>;$
 $PC<15:13> \rightarrow PCLATH<7:5>$

Status Affected: None

Encoding:

110k	kkkk	kkkk	kkkk
------	------	------	------

Description: GOTO allows an unconditional branch anywhere within an 8K page boundary. The thirteen bit immediate value is loaded into PC bits <12:0>. Then the upper eight bits of PC are loaded into PCLATH. GOTO is always a two-cycle instruction.

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: GOTO THERE

After Instruction

PC = Address (THERE)

INFSNZ		Increment f, skip if not 0						
Syntax:	[<i>label</i>] INFSNZ f,d							
Operands:	0 ≤ f ≤ 255 d ∈ [0,1]							
Operation:	(f) + 1 → (dest), skip if not 0							
Status Affected:	None							
Encoding:	<table border="1"><tr><td>0010</td><td>010d</td><td>ffff</td><td>ffff</td></tr></table>				0010	010d	ffff	ffff
0010	010d	ffff	ffff					
Description:	<p>The contents of register 'f' are incremented. If 'd' is 0 the result is placed in WREG. If 'd' is 1 the result is placed back in register 'f'.</p> <p>If the result is not 0, the next instruction, which is already fetched, is discarded, and an NOP is executed instead making it a two-cycle instruction.</p>							
Words:	1							
Cycles:	1(2)							

Q Cycle Activity:

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

If skip:

Q1	Q2	Q3	Q4
Forced NOP	NOP	Execute	NOP

Example: HERE INFSNZ REG, 1
 ZERO
 NZERO

Before Instruction

REG = REG

After Instruction

REG = REG + 1

If REG = 1;

PC = Address (ZERO)

If REG = 0;

PC = Address (NZERO)

IORLW	Inclusive OR Literal with WREG				
Syntax:	[<i>label</i>] IORLW k				
Operands:	$0 \leq k \leq 255$				
Operation:	(WREG) .OR. (k) \rightarrow (WREG)				
Status Affected:	Z				
Encoding:	<table><tr><td>1011</td><td>0011</td><td>kkkk</td><td>kkkk</td></tr></table>	1011	0011	kkkk	kkkk
1011	0011	kkkk	kkkk		
Description:	The contents of WREG are OR'ed with the eight bit literal 'k'. 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: IORLW 0x35

Before Instruction

WREG = 0x9A

After Instruction

WREG = 0xBF

MOVLR	Move Literal to high nibble in BSR			
Syntax:	[<i>label</i>] MOVLR k			
Operands:	$0 \leq k \leq 15$			
Operation:	$k \rightarrow (\text{BSR} \langle 7:4 \rangle)$			
Status Affected:	None			
Encoding:	1011	101x	kkkk	uuuu
Description:	The 4-bit literal 'k' is loaded into the most significant 4-bits of the Bank Select Register (BSR). Only the high 4-bits of the Bank Select Register are affected. The lower half of the BSR is unchanged. The assembler will encode the “u” fields as 0.			
Words:	1			
Cycles:	1			
Q Cycle Activity:				

Q1	Q2	Q3	Q4
Decode	Read literal 'k:u'	Execute	Write literal 'k' to BSR<7:4>

Example: MOVLR 5

Before Instruction

BSR register = 0x22

After Instruction

BSR register = 0x52

Note: This instruction is not available in the PIC17C42 device.

MOVLW	Move Literal to WREG				
Syntax:	[<i>label</i>] MOVLW k				
Operands:	$0 \leq k \leq 255$				
Operation:	$k \rightarrow (\text{WREG})$				
Status Affected:	None				
Encoding:	<table><tr><td>1011</td><td>0000</td><td>kkkk</td><td>kkkk</td></tr></table>	1011	0000	kkkk	kkkk
1011	0000	kkkk	kkkk		
Description:	The eight bit literal 'k' is loaded into WREG.				
Words:	1				
Cycles:	1				
Q Cycle Activity:					

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

Example: MOVLW 0x5A

After Instruction

WREG = 0x5A

PIC17C4X

SUBWF Subtract WREG from f

Syntax: [label] SUBWF f,d

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

Operation: $(f) - (W) \rightarrow (\text{dest})$

Status Affected: OV, C, DC, Z

Encoding:

0000	010d	ffff	ffff
------	------	------	------

Description: Subtract WREG from register 'f' (2's complement method). 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 1: SUBWF REG1, 1

Before Instruction

REG1 = 3
WREG = 2
C = ?

After Instruction

REG1 = 1
WREG = 2
C = 1 ; result is positive
Z = 0

Example 2:

Before Instruction

REG1 = 2
WREG = 2
C = ?

After Instruction

REG1 = 0
WREG = 2
C = 1 ; result is zero
Z = 1

Example 3:

Before Instruction

REG1 = 1
WREG = 2
C = ?

After Instruction

REG1 = FF
WREG = 2
C = 0 ; result is negative
Z = 0

SUBWFB Subtract WREG from f with Borrow

Syntax: [label] SUBWFB f,d

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

Operation: $(f) - (W) - \overline{C} \rightarrow (\text{dest})$

Status Affected: OV, C, DC, Z

Encoding:

0000	001d	ffff	ffff
------	------	------	------

Description: Subtract WREG and the carry flag (borrow) from register 'f' (2's complement method). 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 1: SUBWFB REG1, 1

Before Instruction

REG1 = 0x19 (0001 1001)
WREG = 0x0D (0000 1101)
C = 1

After Instruction

REG1 = 0x0C (0000 1011)
WREG = 0x0D (0000 1101)
C = 1 ; result is positive
Z = 0

Example2: SUBWFB REG1,0

Before Instruction

REG1 = 0x1B (0001 1011)
WREG = 0x1A (0001 1010)
C = 0

After Instruction

REG1 = 0x1B (0001 1011)
WREG = 0x00
C = 1 ; result is zero
Z = 1

Example3: SUBWFB REG1,1

Before Instruction

REG1 = 0x03 (0000 0011)
WREG = 0x0E (0000 1101)
C = 1

After Instruction

REG1 = 0xF5 (1111 0100) [2's comp]
WREG = 0x0E (0000 1101)
C = 0 ; result is negative
Z = 0

PIC17C4X

TABLRD Table Read

Example1: TABLRD 1, 1, REG ;

Before Instruction

REG = 0x53
TBLATH = 0xAA
TBLATL = 0x55
TBLPTR = 0xA356
MEMORY(TBLPTR) = 0x1234

After Instruction (table write completion)

REG = 0xAA
TBLATH = 0x12
TBLATL = 0x34
TBLPTR = 0xA357
MEMORY(TBLPTR) = 0x5678

Example2: TABLRD 0, 0, REG ;

Before Instruction

REG = 0x53
TBLATH = 0xAA
TBLATL = 0x55
TBLPTR = 0xA356
MEMORY(TBLPTR) = 0x1234

After Instruction (table write completion)

REG = 0x55
TBLATH = 0x12
TBLATL = 0x34
TBLPTR = 0xA356
MEMORY(TBLPTR) = 0x1234

TABLWT Table Write

Syntax: [label] TABLWT t,i,f

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

Operation: If $t = 0$,
 $f \rightarrow \text{TBLATL}$;
If $t = 1$,
 $f \rightarrow \text{TBLATH}$;
 $\text{TBLAT} \rightarrow \text{Prog Mem (TBLPTR)}$;
If $i = 1$,
 $\text{TBLPTR} + 1 \rightarrow \text{TBLPTR}$

Status Affected: None

Encoding:

1010	11ti	ffff	ffff
------	------	------	------

Description:

1. Load value in 'f' into 16-bit table latch (TBLAT)
If $t = 0$: load into low byte;
If $t = 1$: load into high byte
2. The contents of TBLAT is written to the program memory location pointed to by TBLPTR
If TBLPTR points to external program memory location, then the instruction takes two-cycle
If TBLPTR points to an internal EPROM location, then the instruction is terminated when an interrupt is received.

Note: The $\overline{\text{MCLR}}$ /VPP pin must be at the programming voltage for successful programming of internal memory.
If $\overline{\text{MCLR}}$ /VPP = VDD
the programming sequence of internal memory will be executed, but will not be successful (although the internal memory location may be disturbed)

3. The TBLPTR can be automatically incremented
If $i = 0$; TBLPTR is not incremented
If $i = 1$; TBLPTR is incremented

Words: 1

Cycles: 2 (many if write is to on-chip EPROM program memory)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Execute	Write register TBLATH or TBLATL

PIC17C4X

Applicable Devices 42 R42 42A 43 R43 44

FIGURE 17-3: CLKOUT AND I/O TIMING

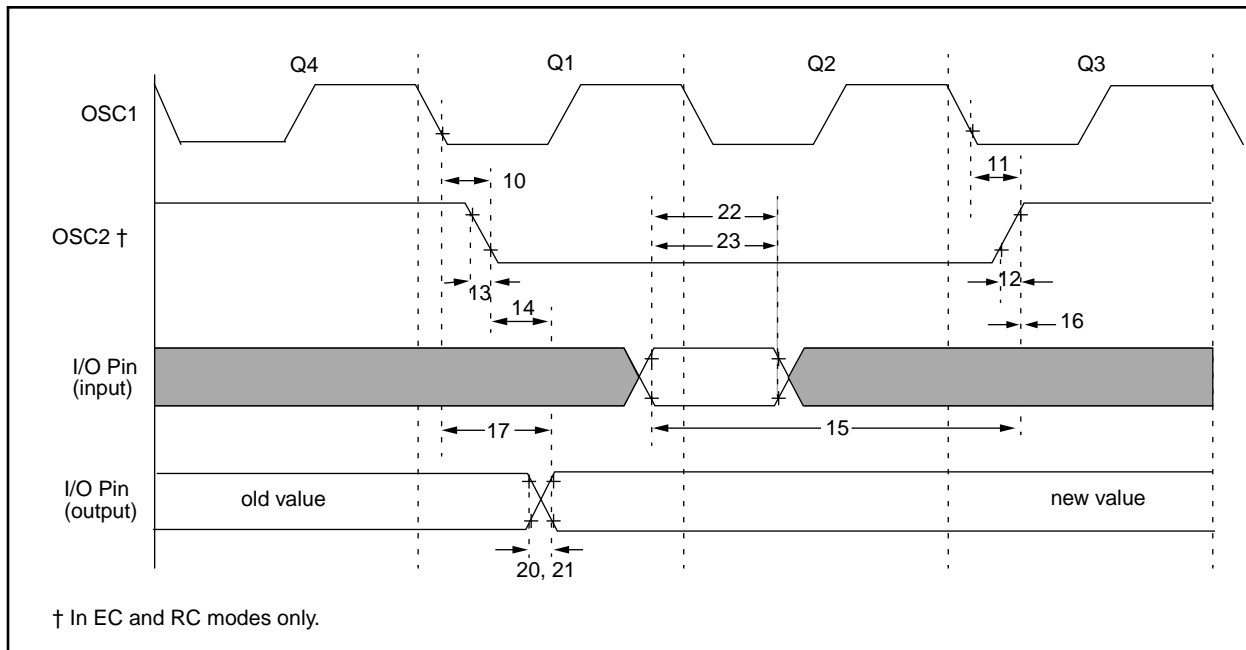


TABLE 17-3: CLKOUT AND I/O TIMING REQUIREMENTS

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	—	—	0.5Tcy + 20‡	ns	Note 1
15	TioV2ckH	Port in valid before CLKOUT↑	0.25Tcy + 25 ‡	—	—	ns	Note 1
16	TckH2ioI	Port in hold after CLKOUT↑	0 ‡	—	—	ns	Note 1
17	TosH2ioV	OSC1↑ (Q1 cycle) to Port out valid	—	—	100 ‡	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 OSC2 output = 4 x TOSC = Tcy.

FIGURE 18-15: I_{OH} vs. V_{OH} , $V_{DD} = 5V$

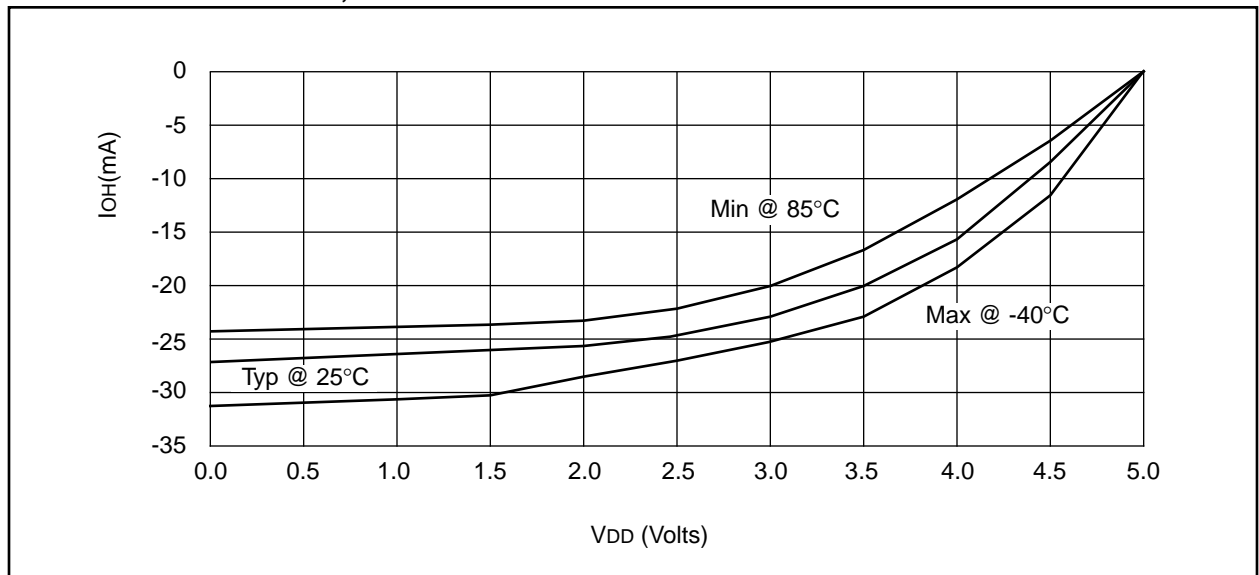
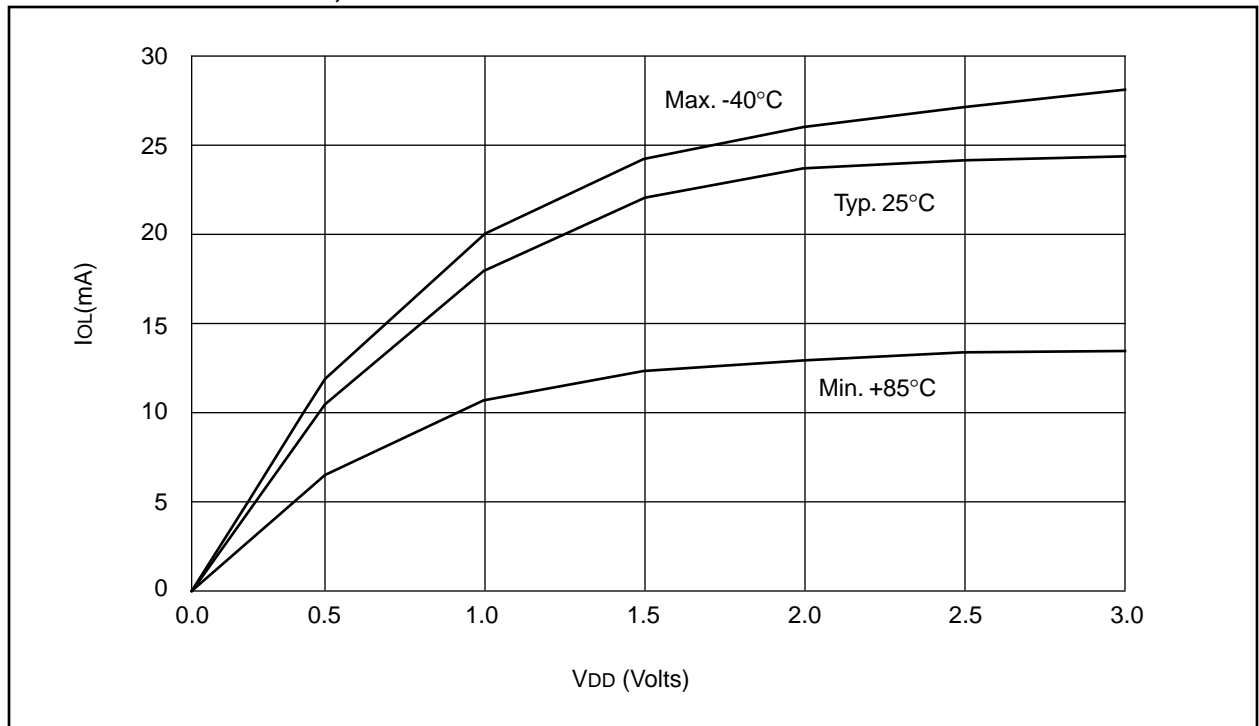


FIGURE 18-16: I_{OL} vs. V_{OL} , $V_{DD} = 3V$



PIC17C4X

Applicable Devices 42 R42 42A 43 R43 44

FIGURE 18-17: I_{OH} vs. V_{OL} , $V_{DD} = 5V$

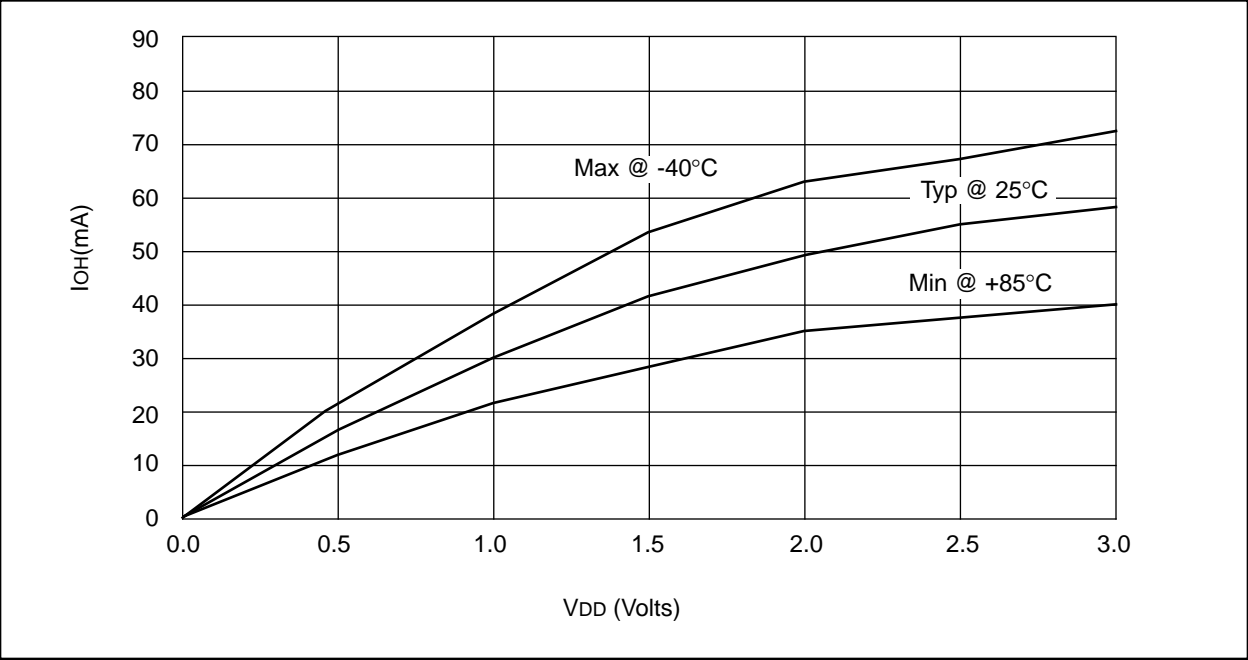


FIGURE 18-18: V_{TH} (INPUT THRESHOLD VOLTAGE) OF I/O PINS (TTL) vs. V_{DD}

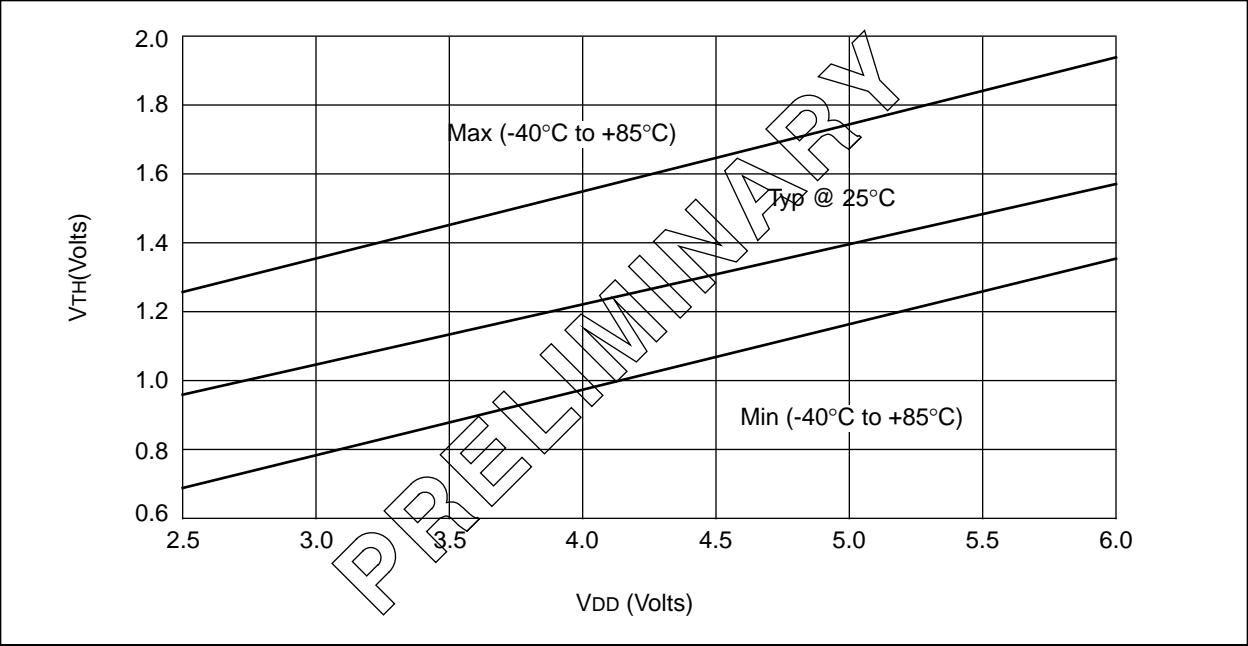


FIGURE 19-5: TIMER0 CLOCK TIMINGS

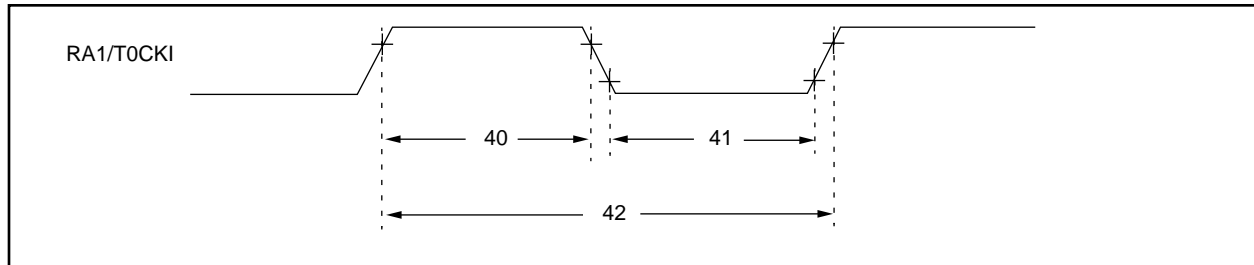


TABLE 19-5: TIMER0 CLOCK REQUIREMENTS

Parameter No.	Sym	Characteristic		Min	Typ†	Max	Units	Conditions
40	Tt0H	T0CKI High Pulse Width	No Prescaler	0.5Tcy + 20 §	—	—	ns	
			With Prescaler	10*	—	—	ns	
41	Tt0L	T0CKI Low Pulse Width	No Prescaler	0.5Tcy + 20 §	—	—	ns	
			With Prescaler	10*	—	—	ns	
42	Tt0P	T0CKI Period		Greater of: 20 ns or $\frac{Tcy + 40 §}{N}$	—	—	ns	N = prescale value (1, 2, 4, ..., 256)

* 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 19-6: TIMER1, TIMER2, AND TIMER3 CLOCK TIMINGS

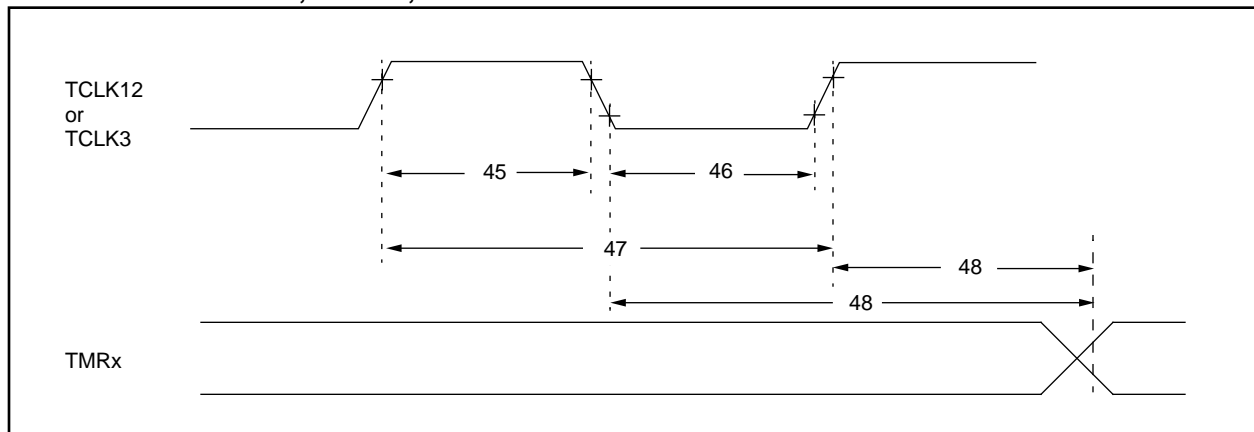


TABLE 19-6: TIMER1, TIMER2, AND TIMER3 CLOCK REQUIREMENTS

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
45	Tt123H	TCLK12 and TCLK3 high time	0.5Tcy + 20 §	—	—	ns	
46	Tt123L	TCLK12 and TCLK3 low time	0.5Tcy + 20 §	—	—	ns	
47	Tt123P	TCLK12 and TCLK3 input period	$\frac{Tcy + 40 §}{N}$	—	—	ns	N = prescale value (1, 2, 4, 8)
48	TckE2tmr1	Delay from selected External Clock Edge to Timer increment	2Tosc §		6Tosc §		

* 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.

PIC17C4X

Applicable Devices 42 R42 42A 43 R43 44

FIGURE 20-2: TYPICAL RC OSCILLATOR FREQUENCY vs. VDD

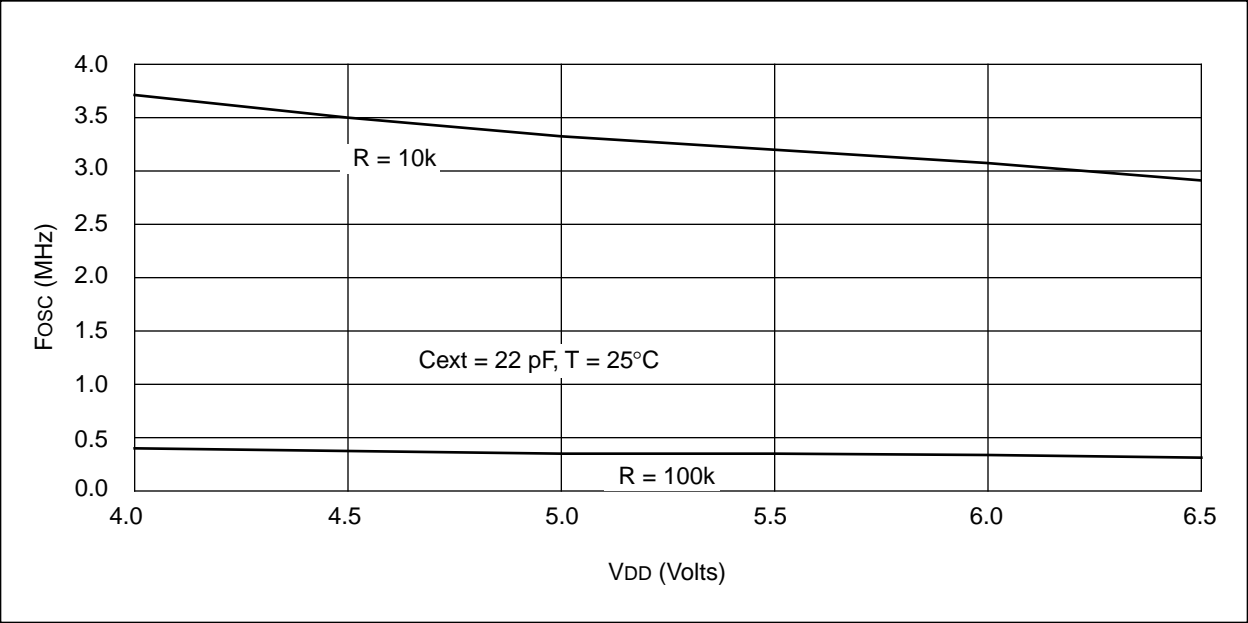
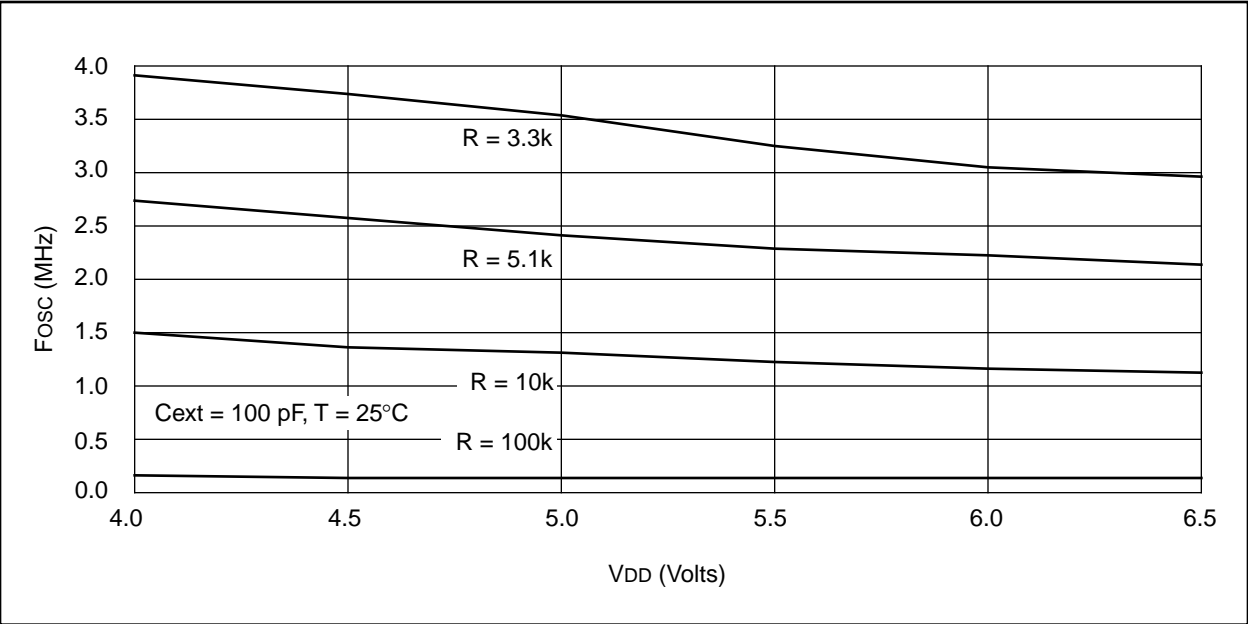


FIGURE 20-3: TYPICAL RC OSCILLATOR FREQUENCY vs. VDD



PIC17C4X

Applicable Devices 42 R42 42A 43 R43 44

FIGURE 20-13: WDT TIMER TIME-OUT PERIOD vs. VDD

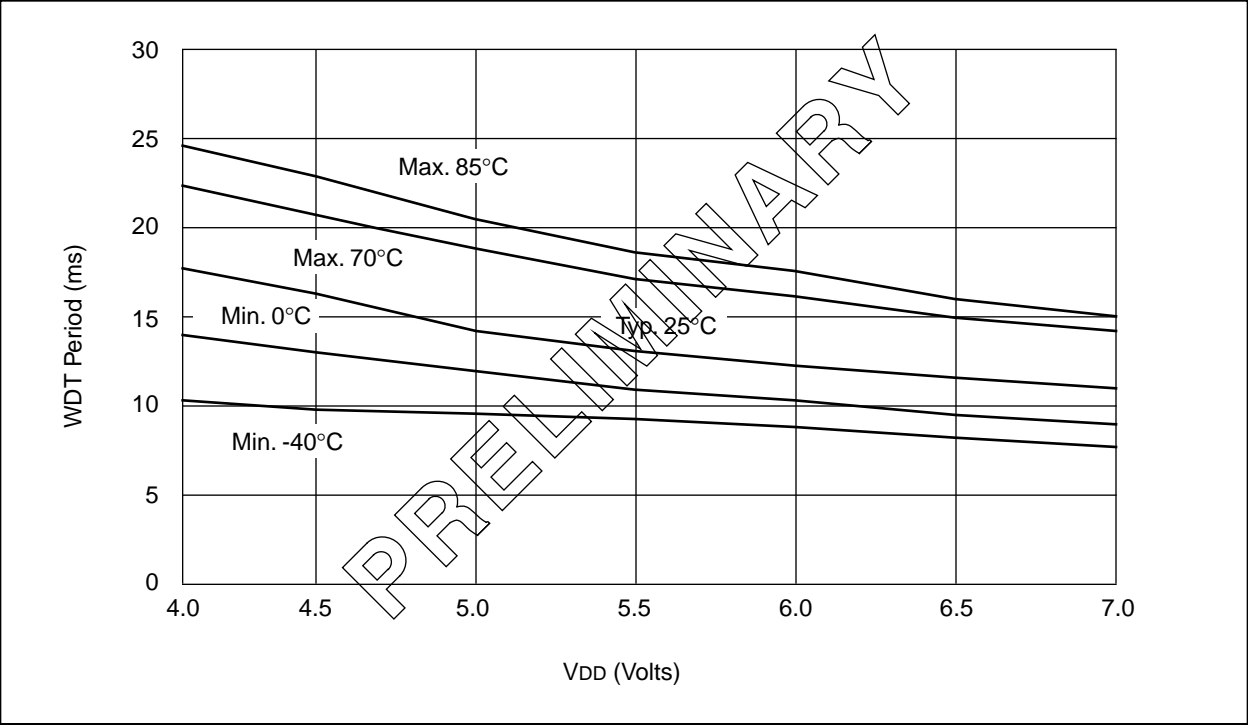
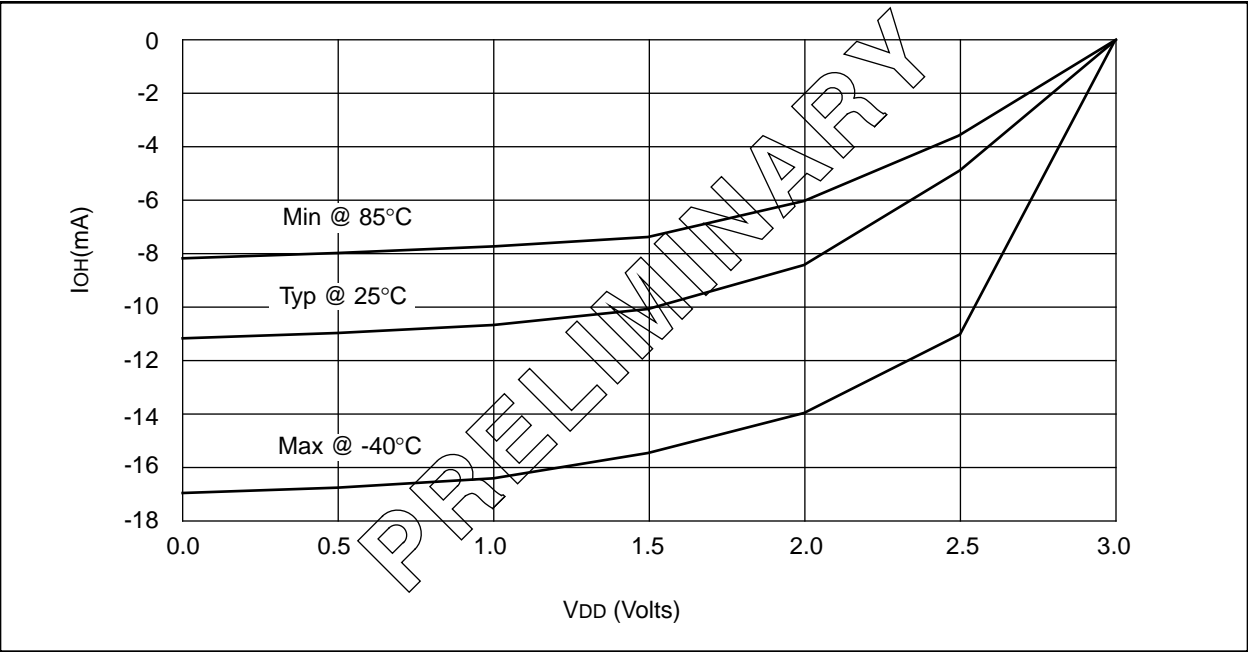


FIGURE 20-14: IOH vs. VOH, VDD = 3V



PIC17C4X

E.2 PIC16C5X Family of Devices

	Clock		Memory		Peripherals		Features	
	Maximum Frequency of Operation (MHz)	Program Memory (x12 words)	ROM	RAM Data Memory (bytes)	Timer Module(s)	I/O Pins	Voltage Range (Volts)	Number of Instructions
PIC16C52	4	384	—	25	TMR0	12	2.5-6.25	33
PIC16C54	20	512	—	25	TMR0	12	2.5-6.25	33
PIC16C54A	20	512	—	25	TMR0	12	2.0-6.25	33
PIC16CR54A	20	—	512	25	TMR0	12	2.0-6.25	33
PIC16C55	20	512	—	24	TMR0	20	2.5-6.25	33
PIC16C56	20	1K	—	25	TMR0	12	2.5-6.25	33
PIC16C57	20	2K	—	72	TMR0	20	2.5-6.25	33
PIC16CR57B	20	—	2K	72	TMR0	20	2.5-6.25	33
PIC16C58A	20	2K	—	73	TMR0	12	2.0-6.25	33
PIC16CR58A	20	—	2K	73	TMR0	12	2.5-6.25	33

All PIC16/17 Family devices have Power-On Reset, selectable Watchdog Timer, selectable code protect and high I/O current capability.

PIC17C4X

E.8 PIC17CXX Family of Devices

	Clock			Memory			Peripherals				Features		
	Maximum Frequency of Operation (MHz)	ROM	Program Memory (Words)	RAM Data Memory (bytes)	Timer Module(s)	Captures/PWMs	Serial Port(s) (USART)	Hardware Multiply	External Interrupts	Interrupt Sources	I/O Pins	Voltage Range (Volts)	Packages
PIC17C42	25	2K	—	232	TMR0, TMR1, TMR2, TMR3	2 2	Yes	—	Yes	11	33	4.5-5.5	40-pin DIP; 44-pin PLCC, MQFP
PIC17C42A	25	2K	—	232	TMR0, TMR1, TMR2, TMR3	2 2	Yes	Yes	Yes	11	33	2.5-6.0	40-pin DIP; 44-pin PLCC, TQFP, MQFP
PIC17CR42	25	—	2K	232	TMR0, TMR1, TMR2, TMR3	2 2	Yes	Yes	Yes	11	33	2.5-6.0	40-pin DIP; 44-pin PLCC, TQFP, MQFP
PIC17C43	25	4K	—	454	TMR0, TMR1, TMR2, TMR3	2 2	Yes	Yes	Yes	11	33	2.5-6.0	40-pin DIP; 44-pin PLCC, TQFP, MQFP
PIC17CR43	25	—	4K	454	TMR0, TMR1, TMR2, TMR3	2 2	Yes	Yes	Yes	11	33	2.5-6.0	40-pin DIP; 44-pin PLCC, TQFP, MQFP
PIC17C44	25	8K	—	454	TMR0, TMR1, TMR2, TMR3	2 2	Yes	Yes	Yes	11	33	2.5-6.0	40-pin DIP; 44-pin PLCC, TQFP, MQFP

All PIC16/17 Family devices have Power-on Reset, selectable Watchdog Timer, selectable code protect and high I/O current capability.