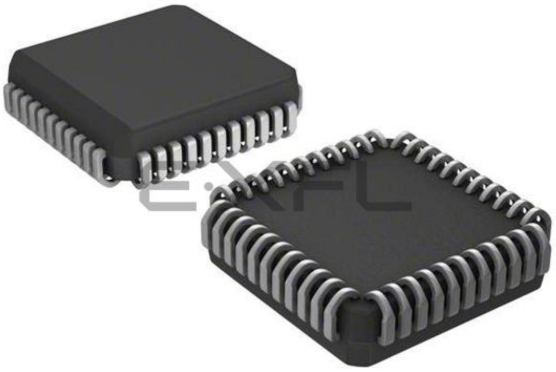


Welcome to E-XFL.COM

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	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 ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	44-LCC (J-Lead)
Supplier Device Package	44-PLCC (16.59x16.59)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic17c44-25i-l

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.

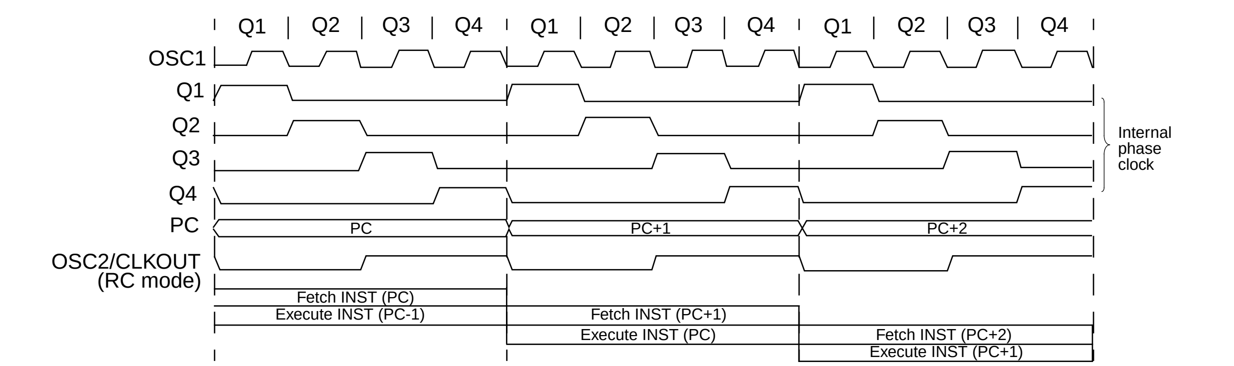
3.2 Instruction Flow/Pipelining

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

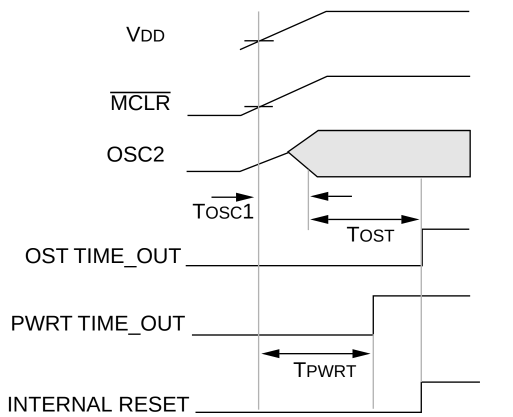


EXAMPLE 3-2: INSTRUCTION PIPELINE FLOW

	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

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.

FIGURE 4-5: OSCILLATOR START-UP TIME



This figure shows in greater detail the timings involved with the oscillator start-up timer. In this example the low frequency crystal start-up time is larger than power-up time (TPWRT).
Tosc1 = time for the crystal oscillator to react to an oscillation level detectable by the Oscillator Start-up Timer (ost).
TOST = 1024TOSC.

FIGURE 4-6: USING ON-CHIP POR

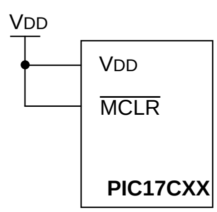
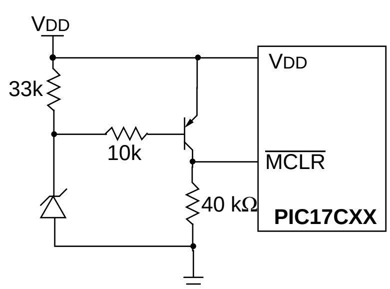
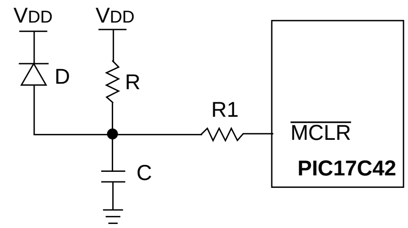


FIGURE 4-7: BROWN-OUT PROTECTION CIRCUIT 1



This circuit will activate reset when VDD goes below (Vz + 0.7V) where Vz = Zener voltage.

FIGURE 4-8: PIC17C42 EXTERNAL POWER-ON RESET CIRCUIT (FOR SLOW VDD POWER-UP)

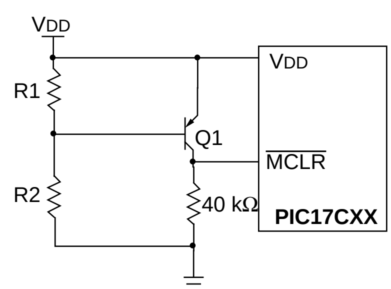


Note 1: An external Power-on Reset circuit is required only if VDD power-up time is too slow. The diode D helps discharge the capacitor quickly when VDD powers down.

2: R < 40 kΩ is recommended to ensure that the voltage drop across R does not exceed 0.2V (max. leakage current spec. on the MCLR/VPP pin is 5 μA). A larger voltage drop will degrade VIH level on the MCLR/VPP pin.

3: R1 = 100Ω to 1 kΩ will limit any current flowing into MCLR from external capacitor C in the event of MCLR/VPP pin breakdown due to Electrostatic Discharge (ESD) or (Electrical Overstress) EOS.

FIGURE 4-9: BROWN-OUT PROTECTION CIRCUIT 2



This brown-out circuit is less expensive, albeit less accurate. Transistor Q1 turns off when VDD is below a certain level such that:

$$V_{DD} \bullet \frac{R1}{R1 + R2} = 0.7V$$

Example 8-3 shows the sequence to do a 16 x 16 unsigned multiply. Equation 8-1 shows the algorithm that is used. The 32-bit result is stored in 4 registers RES3:RES0.

EQUATION 8-1: 16 x 16 UNSIGNED MULTIPLICATION ALGORITHM

RES3:RES0 = ARG1H:ARG1L * ARG2H:ARG2L

= (ARG1H * ARG2H * 2^{16}) +

(ARG1H * ARG2L * 2^{8}) +

(ARG1L * ARG2H * 2^{8}) +

(ARG1L * ARG2L)

EXAMPLE 8-3: 16 x 16 MULTIPLY ROUTINE

```
   MOVFP    ARG1L, WREG
   MULWF    ARG2L        ; ARG1L * ARG2L ->
                         ;   PRODH:PRODL
   MOVPF    PRODH, RES1 ;
   MOVPF    PRODL, RES0 ;
;
   MOVFP    ARG1H, WREG
   MULWF    ARG2H        ; ARG1H * ARG2H ->
                         ;   PRODH:PRODL
   MOVPF    PRODH, RES3 ;
   MOVPF    PRODL, RES2 ;
;
   MOVFP    ARG1L, WREG
   MULWF    ARG2H        ; ARG1L * ARG2H ->
                         ;   PRODH:PRODL
   MOVFP    PRODL, WREG ;
   ADDWF    RES1, F      ; Add cross
   MOVFP    PRODH, WREG ;    products
   ADDWFC   RES2, F      ;
   CLRF     WREG, F      ;
   ADDWFC   RES3, F      ;
;
   MOVFP    ARG1H, WREG ;
   MULWF    ARG2L        ; ARG1H * ARG2L ->
                         ;   PRODH:PRODL

   MOVFP    PRODL, WREG ;
   ADDWF    RES1, F      ; Add cross
   MOVFP    PRODH, WREG ;    products
   ADDWFC   RES2, F      ;
   CLRF     WREG, F      ;
   ADDWFC   RES3, F      ;
```

Example 8-4 shows the sequence to do an 16 x 16 signed multiply. Equation 8-2 shows the algorithm that used. The 32-bit result is stored in four registers RES3:RES0. To account for the sign bits of the arguments, each argument pairs most significant bit (MSb) is tested and the appropriate subtractions are done.

EQUATION 8-2: 16 x 16 SIGNED MULTIPLICATION ALGORITHM

RES3:RES0

$$= ARG1H:ARG1L * ARG2H:ARG2L$$

$$= (ARG1H * ARG2H * 2^{16}) +$$
$$(ARG1H * ARG2L * 2^{8}) +$$
$$(ARG1L * ARG2H * 2^{8}) +$$
$$(ARG1L * ARG2L) +$$
$$(-1 * ARG2H<7> * ARG1H:ARG1L * 2^{16}) +$$
$$(-1 * ARG1H<7> * ARG2H:ARG2L * 2^{16})$$

EXAMPLE 8-4: 16 x 16 SIGNED MULTIPLY ROUTINE

```
   MOVFP    ARG1L, WREG
   MULWF    ARG2L       ; ARG1L * ARG2L ->
                        ;   PRODH:PRODL
   MOVPF    PRODH, RES1 ;
   MOVPF    PRODL, RES0 ;
;
   MOVFP    ARG1H, WREG
   MULWF    ARG2H       ; ARG1H * ARG2H ->
                        ;   PRODH:PRODL
   MOVPF    PRODH, RES3 ;
   MOVPF    PRODL, RES2 ;
;
   MOVFP    ARG1L, WREG
   MULWF    ARG2H       ; ARG1L * ARG2H ->
                        ;   PRODH:PRODL
   MOVFP    PRODL, WREG ;
   ADDWF    RES1, F     ; Add cross
   MOVFP    PRODH, WREG ;    products
   ADDWFC   RES2, F     ;
   CLRF     WREG, F     ;
   ADDWFC   RES3, F     ;
;
   MOVFP    ARG1H, WREG ;
   MULWF    ARG2L       ; ARG1H * ARG2L ->
                        ;   PRODH:PRODL

   MOVFP    PRODL, WREG ;
   ADDWF    RES1, F     ; Add cross
   MOVFP    PRODH, WREG ;    products
   ADDWFC   RES2, F     ;
   CLRF     WREG, F     ;
   ADDWFC   RES3, F     ;
;
   BTFSS    ARG2H, 7    ; ARG2H:ARG2L neg?
   GOTO     SIGN_ARG1   ; no, check ARG1
   MOVFP    ARG1L, WREG ;
   SUBWF    RES2        ;
   MOVFP    ARG1H, WREG ;
   SUBWFB   RES3
;
SIGN_ARG1
   BTFSS    ARG1H, 7    ; ARG1H:ARG1L neg?
   GOTO     CONT_CODE   ; no, done
   MOVFP    ARG2L, WREG ;
   SUBWF    RES2        ;
   MOVFP    ARG2H, WREG ;
   SUBWFB   RES3
;
CONT_CODE
      :
```

FIGURE 11-5: TMR0 READ/WRITE IN TIMER MODE

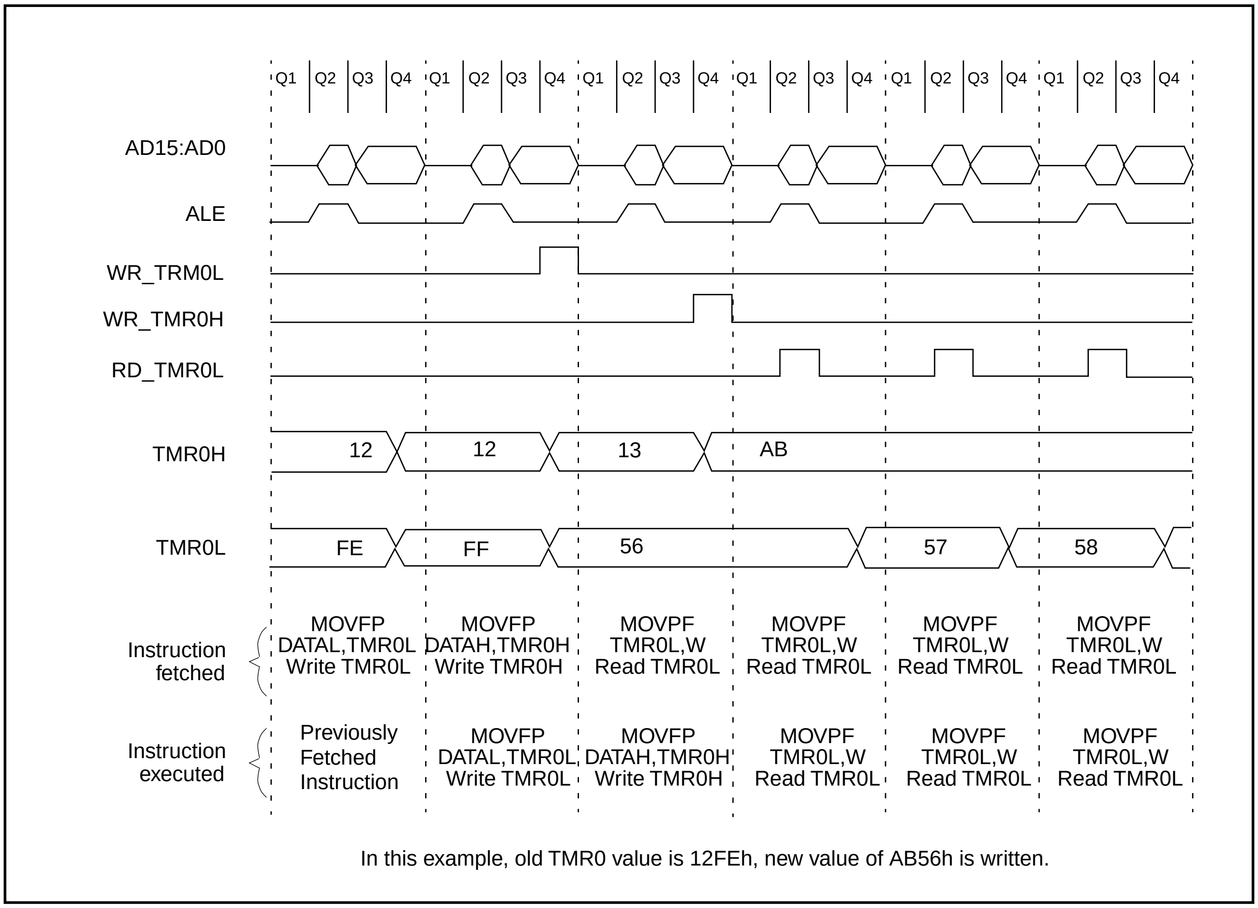


TABLE 11-1: REGISTERS/BITS ASSOCIATED WITH TIMER0

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)
05h, Unbanked	T0STA	INTEDG	T0SE	T0CS	PS3	PS2	PS1	PS0	—	0000 000-	0000 000-
06h, Unbanked	CPUSTA	—	—	STKAV	GLINTD	$\overline{\text{TO}}$	$\overline{\text{PD}}$	—	—	--11 11--	--11 qq--
07h, Unbanked	INTSTA	PEIF	T0CKIF	T0IF	INTF	PEIE	T0CKIE	T0IE	INTE	0000 0000	0000 0000
0Bh, Unbanked	TMR0L	TMR0 register; low byte								xxxx xxxx	uuuu uuuu
0Ch, Unbanked	TMR0H	TMR0 register; high byte								xxxx xxxx	uuuu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented read as a '0', q - value depends on condition, Shaded cells are not used by Timer0.
Note 1: Other (non power-up) resets include: external reset through $\overline{\text{MCLR}}$ and the Watchdog Timer Reset.

13.2 USART Asynchronous Mode

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

The asynchronous mode is selected by clearing the SYNC bit (TXSTA<4>).

The USART Asynchronous module consists of the following important elements:

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

13.2.1 USART ASYNCHRONOUS TRANSMITTER

The USART transmitter block diagram is shown in Figure 13-3. The heart of the transmitter is the transmit 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 stop bit has been transmitted from the previous load. As soon as the stop bit is transmitted, the TSR is loaded with new data from the TXREG (if available). Once TXREG transfers the data to the TSR (occurs in one T_{CY} at the end of the current BRG cycle), the TXREG is empty and an interrupt bit, TXIF (PIR<1>) is set. This interrupt can be enabled or disabled by the TXIE bit (PIE<1>). TXIF will be set regardless of TXIE and cannot be reset in software. It will reset only when new data is loaded into TXREG. While TXIF indicates the status of the TXREG, the TRMT (TXSTA<1>) bit 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.

Note:	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 and the baud rate generator (BRG) has produced a shift clock (Figure 13-5). The transmission can also be started by first loading TXREG and then setting TXEN. Normally when transmission is first started, the TSR is empty, so a transfer to TXREG will result in an immediate transfer to TSR resulting in an empty TXREG. A back-to-back transfer is thus possible (Figure 13-6). Clearing TXEN during a transmission will cause the transmission to be aborted. This will reset the transmitter and the RA5/TX/CK pin will revert to hi-impedance.

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 the 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).

Steps to follow when setting up an Asynchronous Transmission:

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

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

ANDWF — AND WREG with f

Syntax: [*label*] ANDWF f,d

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

Operation: (WREG) .AND. (f) → (dest)

Status Affected: Z

Encoding:

0000	101d	ffff	ffff

Description: The contents of WREG are AND'ed with 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: `ANDWF REG, 1`

Before Instruction
WREG = 0x17
REG = 0xC2

After Instruction
WREG = 0x17
REG = 0x02

BCF — Bit Clear f

Syntax: [*label*] BCF f,b

Operands: $0 \le f \le 255$
$0 \le b \le 7$

Operation: 0 → (f<b>)

Status Affected: None

Encoding:

1000	1bbb	ffff	ffff

Description: Bit 'b' in register 'f' is cleared.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: `BCF FLAG_REG, 7`

Before Instruction
FLAG_REG = 0xC7

After Instruction
FLAG_REG = 0x47

CLRWDT — Clear Watchdog Timer

Syntax: [*label*] CLRWDT

Operands: None

Operation: 00h → WDT
0 → WDT postscaler,
1 → $\overline{TO}$
1 → $\overline{PD}$

Status Affected: $\overline{TO}$, $\overline{PD}$

Encoding:

0000	0000	0000	0100

Description: CLRWDT instruction resets the watchdog timer. It also resets the prescaler of the WDT. Status bits $\overline{TO}$ and $\overline{PD}$ are set.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register ALUSTA	Execute	NOP

Example: CLRWDT

Before Instruction
WDT counter = ?

After Instruction
WDT counter = 0x00
WDT Postscaler = 0
$\overline{TO}$ = 1
$\overline{PD}$ = 1

COMF — Complement f

Syntax: [*label*] COMF f,d

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

Operation: $(\overline{f}) \rightarrow$ (dest)

Status Affected: Z

Encoding:

0001	001d	ffff	ffff

Description: The contents of register 'f' are complemented. 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 register 'f'

Example: COMF REG1,0

Before Instruction
REG1 = 0x13

After Instruction
REG1 = 0x13
WREG = 0xEC

CPFSLT — Compare f with WREG, skip if f < WREG

Syntax: [*label*] CPFSLT f

Operands: $0 \le f \le 255$

Operation: (f) – (WREG),
skip if (f) < (WREG)
(unsigned comparison)

Status Affected: None

Encoding:

`0011`	`0000`	`ffff`	`ffff`

Description: Compares the contents of data memory location 'f' to the contents of WREG by performing an unsigned subtraction.

If the contents of 'f' < the contents of WREG, then the fetched instruction is discarded and an NOP is executed instead making this a two-cycle instruction.

Words: 1

Cycles: 1 (2)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Execute	NOP

If skip:

Q1	Q2	Q3	Q4
Forced NOP	NOP	Execute	NOP

Example:

```
HERE     CPFSLT REG
NLESS    :
LESS     :
```

Before Instruction
PC = Address (`HERE`)
W = ?

After Instruction
If REG < WREG;
PC = Address (`LESS`)
If REG ≥ WREG;
PC = Address (`NLESS`)

DAW — Decimal Adjust WREG Register

Syntax: [*label*] DAW f,s

Operands: $0 \le f \le 255$
$s \in [0,1]$

Operation: If [WREG<3:0> >9] .OR. [DC = 1] then
WREG<3:0> + 6 → f<3:0>, s<3:0>;
else
WREG<3:0> → f<3:0>, s<3:0>;

If [WREG<7:4> >9] .OR. [C = 1] then
WREG<7:4> + 6 → f<7:4>, s<7:4>
else
WREG<7:4> → f<7:4>, s<7:4>

Status Affected: C

Encoding:

`0010`	`111s`	`ffff`	`ffff`

Description: DAW adjusts the eight bit value in WREG resulting from the earlier addition of two variables (each in packed BCD format) and produces a correct packed BCD result.

s = 0: Result is placed in Data memory location 'f' and WREG.

s = 1: Result is placed in Data memory location 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example1: `DAW REG1, 0`

Before Instruction
WREG = 0xA5
REG1 = ??
C = 0
DC = 0

After Instruction
WREG = 0x05
REG1 = 0x05
C = 1
DC = 0

Example 2:

Before Instruction
WREG = 0xCE
REG1 = ??
C = 0
DC = 0

After Instruction
WREG = 0x24
REG1 = 0x24
C = 1
DC = 0

INCF — Increment f

Syntax: [*label*] INCF f,d

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

Operation: (f) + 1 → (dest)

Status Affected: OV, C, DC, Z

Encoding:

`0001`	`010d`	`ffff`	`ffff`

Description: 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'.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: `INCF CNT, 1`

Before Instruction
CNT = 0xFF
Z = 0
C = ?

After Instruction
CNT = 0x00
Z = 1
C = 1

INCFSZ — Increment f, skip if 0

Syntax: [*label*] INCFSZ f,d

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

Operation: (f) + 1 → (dest)
skip if result = 0

Status Affected: None

Encoding:

`0001`	`111d`	`ffff`	`ffff`

Description: 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'.

If the result is 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     INCFSZ   CNT, 1
NZERO    :
ZERO     :
```

Before Instruction
PC = Address (`HERE`)

After Instruction
CNT = CNT + 1
If CNT = 0;
PC = Address(`ZERO`)
If CNT ≠ 0;
PC = Address(`NZERO`)

MOVLR — Move Literal to high nibble in BSR

Syntax:	[*label*] MOVLR k
Operands:	0 ≤ k ≤ 15
Operation:	k → (BSR<7:4>)
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:	[*label*] MOVLW k
Operands:	0 ≤ k ≤ 255
Operation:	k → (WREG)
Status Affected:	None
Encoding:	`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

Applicable Devices	42	R42	42A	43	R43	44

17.4 Timing Diagrams and Specifications

FIGURE 17-2: EXTERNAL CLOCK TIMING

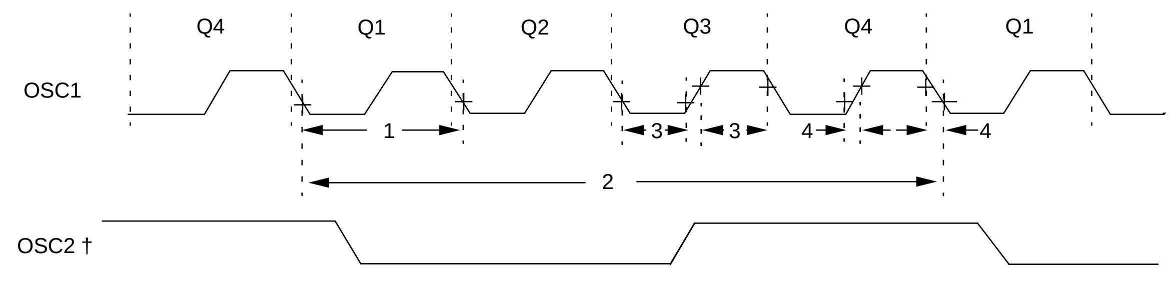


TABLE 17-2: EXTERNAL CLOCK TIMING REQUIREMENTS

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
	Fosc	**External CLKIN Frequency (Note 1)**	DC DC	— —	16 25	MHz MHz	EC osc mode - PIC17C42-16 - PIC17C42-25
		Oscillator Frequency (Note 1)	DC 1 1 DC	— — — —	4 16 25 2	MHz MHz MHz MHz	RC osc mode XT osc mode - PIC17C42-16 - PIC17C42-25 LF osc mode
1	Tosc	**External CLKIN Period (Note 1)**	62.5 40	— —	— —	ns ns	EC osc mode - PIC17C42-16 - PIC17C42-25
		Oscillator Period (Note 1)	250 62.5 40 500	— — — —	— 1,000 1,000 —	ns ns ns ns	RC osc mode XT osc mode - PIC17C42-16 - PIC17C42-25 LF osc mode
2	TCY	**Instruction Cycle Time (Note 1)**	160	4/Fosc	DC	ns	
3	TosL, TosH	**Clock in (OSC1) High or Low Time**	10 ‡	—	—	ns	EC oscillator
4	TosR, TosF	**Clock in (OSC1) Rise or Fall Time**	—	—	5 ‡	ns	EC oscillator

† 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: Instruction cycle period (TCY) equals four times the input oscillator time-base period. All specified values are based on characterization data for that particular oscillator type under standard operating conditions with the device executing code. Exceeding these specified limits may result in unstable oscillator operation and/or higher than expected current consumption. All devices are tested to operate at “min.” values with an external clock applied to the OSC1 pin.
When an external clock input is used, the “Max.” cycle time limit is “DC” (no clock) for all devices.

Applicable Devices	42	R42	42A	43	R43	44

FIGURE 18-17: IOL VS. VOL, VDD = 5V

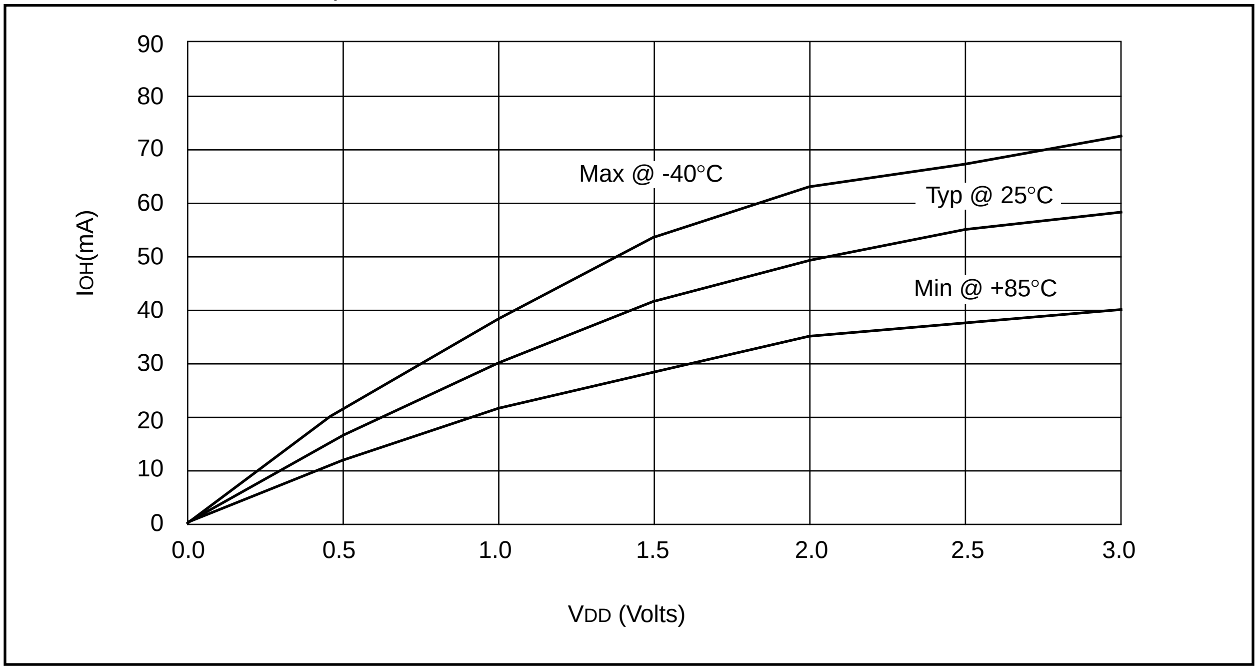
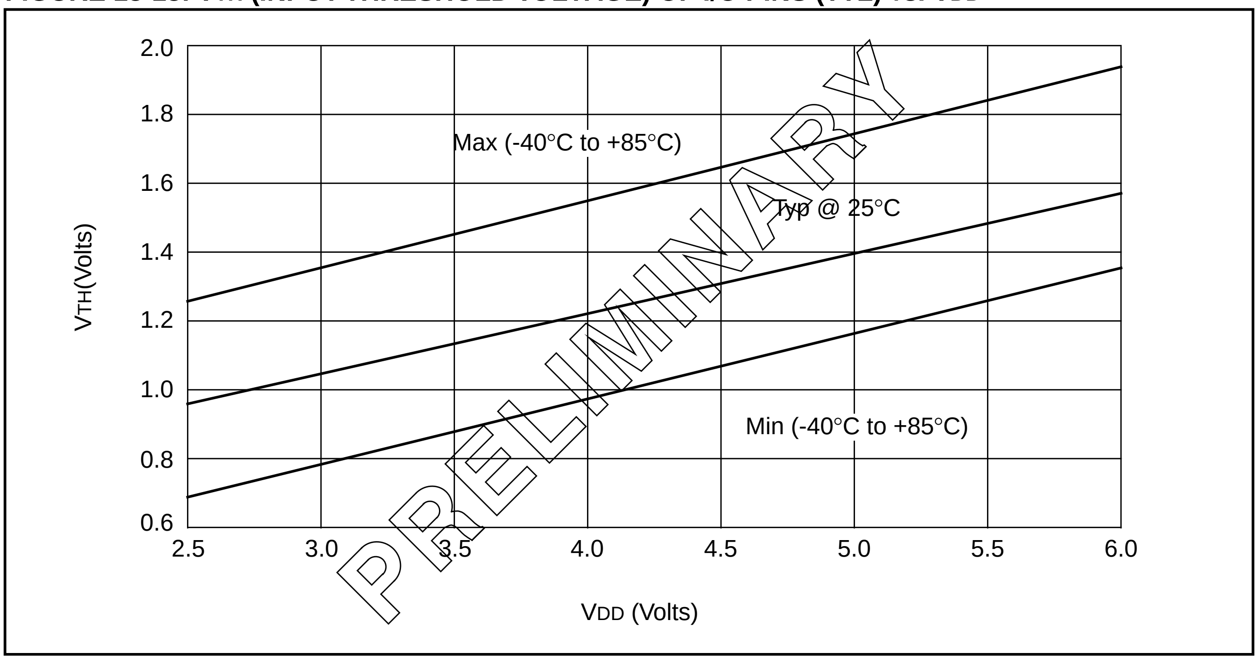


FIGURE 18-18: VTH (INPUT THRESHOLD VOLTAGE) OF I/O PINS (TTL) VS. VDD



Applicable Devices	42	R42	42A	43	R43	44

FIGURE 19-5: TIMER0 CLOCK TIMINGS

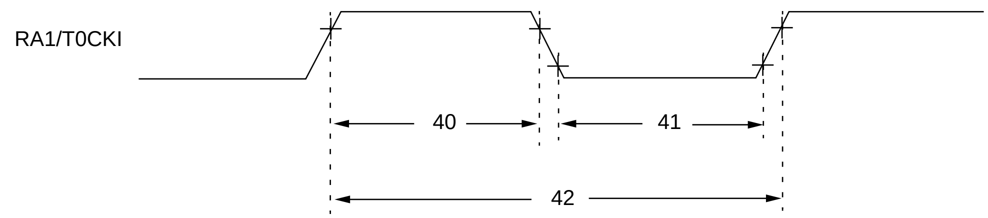


TABLE 19-5: TIMER0 CLOCK REQUIREMENTS

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 (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

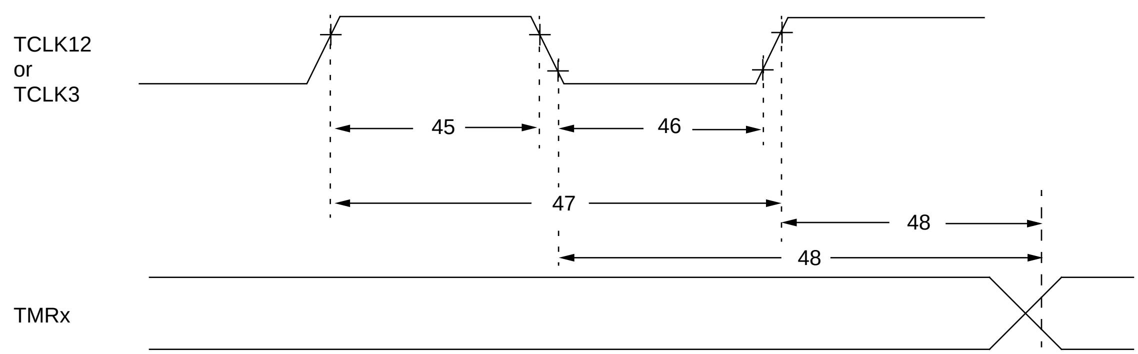


TABLE 19-6: TIMER1, TIMER2, AND TIMER3 CLOCK REQUIREMENTS

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	(TCY + 40 §)/N	—	—	ns	N = prescale value (1, 2, 4, 8)
48	TckE2tmrI	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.

Applicable Devices	42	R42	42A	43	R43	44

FIGURE 19-11: MEMORY INTERFACE WRITE TIMING (NOT SUPPORTED IN PIC17LC4X DEVICES)

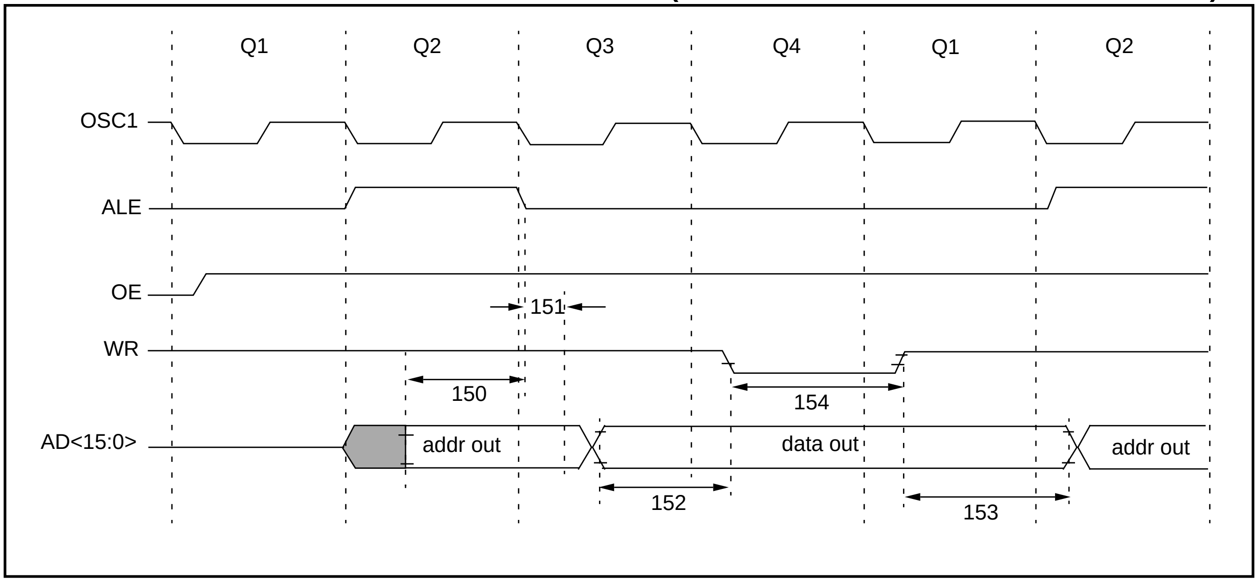


TABLE 19-11: MEMORY INTERFACE WRITE REQUIREMENTS (NOT SUPPORTED IN PIC17LC4X DEVICES)

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
150	TadV2alL	AD<15:0> (address) valid to ALE↓ (address setup time)	0.25Tcy - 10	—	—	ns	
151	TalL2adI	ALE↓ to address out invalid (address hold time)	0	—	—	ns	
152	TadV2wrL	Data out valid to $\overline{WR}$↓ (data setup time)	0.25Tcy - 40	—	—	ns	
153	TwrH2adI	$\overline{WR}$↑ to data out invalid (data hold time)	—	0.25TCY §	—	ns	
154	TwrL	$\overline{WR}$ pulse width	—	0.25TCY §	—	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.

Applicable Devices	42	R42	42A	43	R43	44

FIGURE 20-9: TYPICAL IPD vs. VDD WATCHDOG DISABLED 25°C

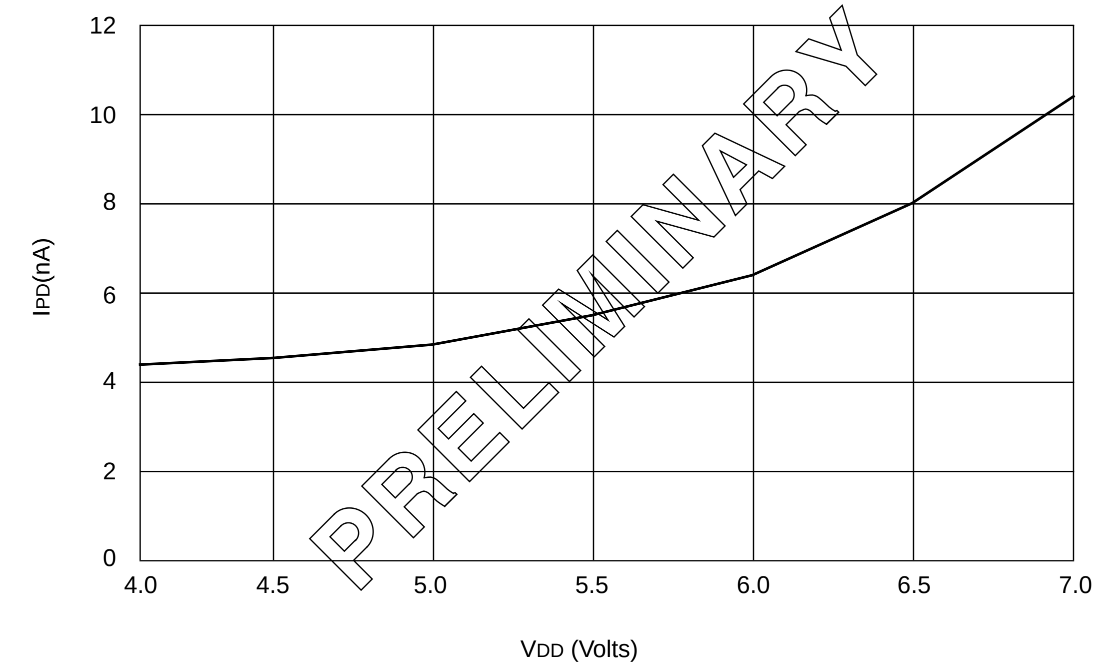
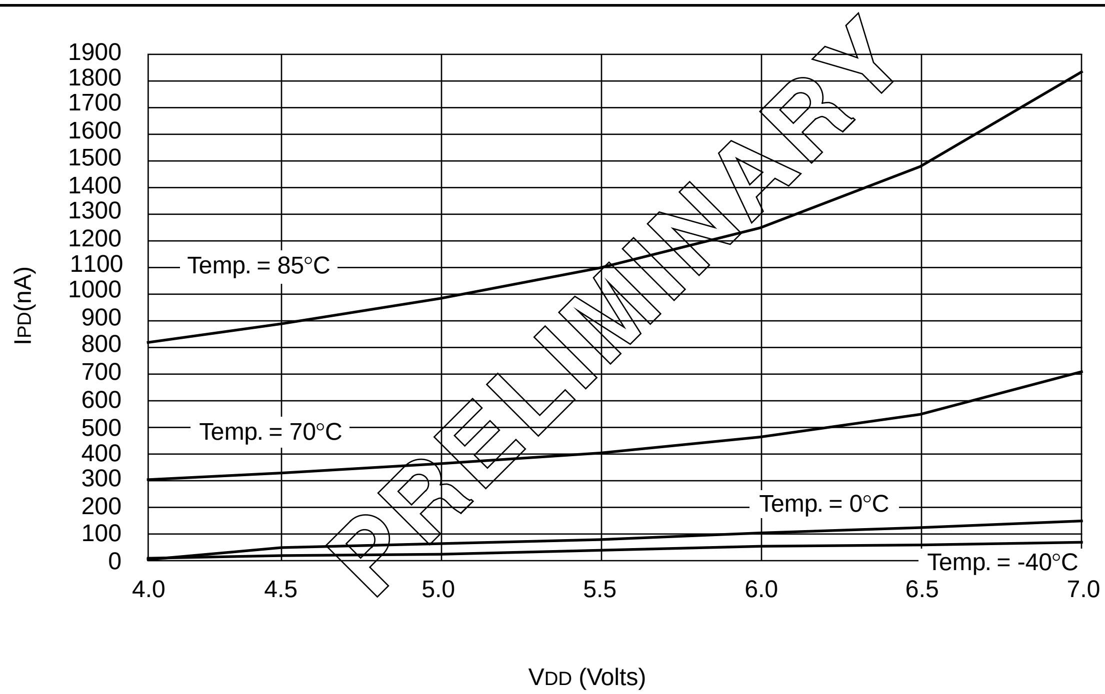


FIGURE 20-10: MAXIMUM IPD vs. VDD WATCHDOG DISABLED



Applicable Devices	42	R42	42A	43	R43	44

FIGURE 20-19: V_{TH}, V_{IL} of I/O PINS (SCHMITT TRIGGER) vs. V_{DD}

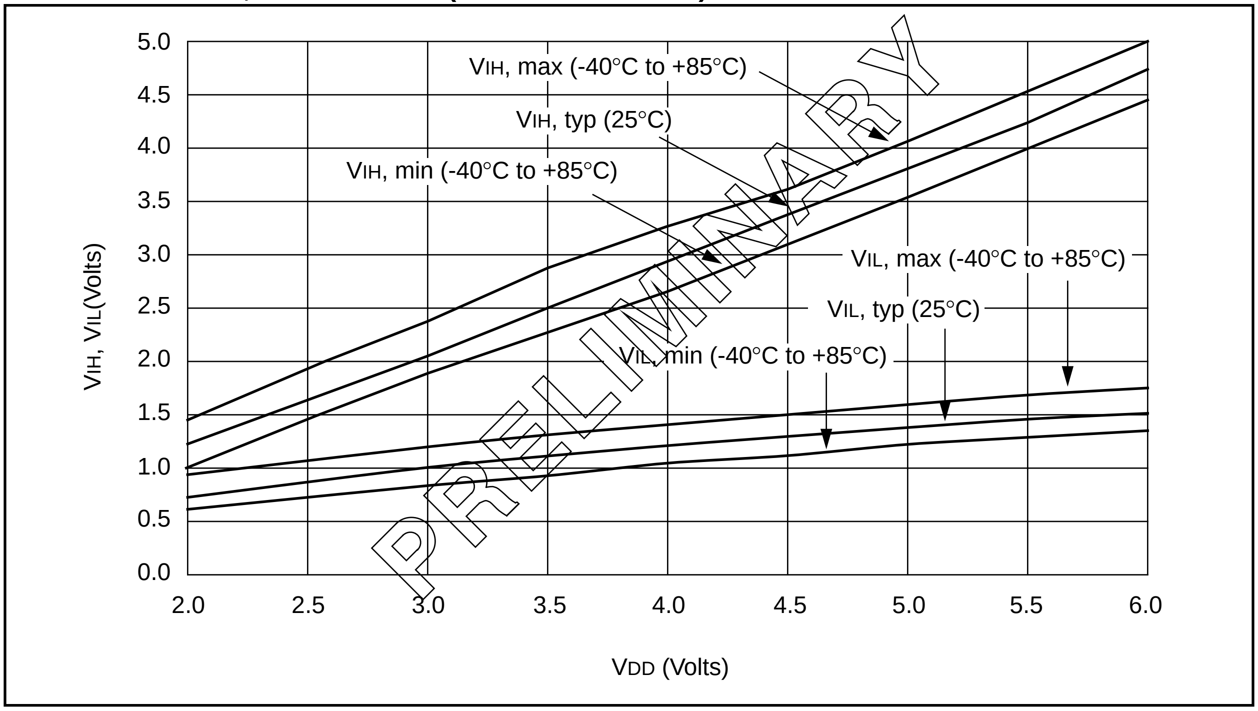
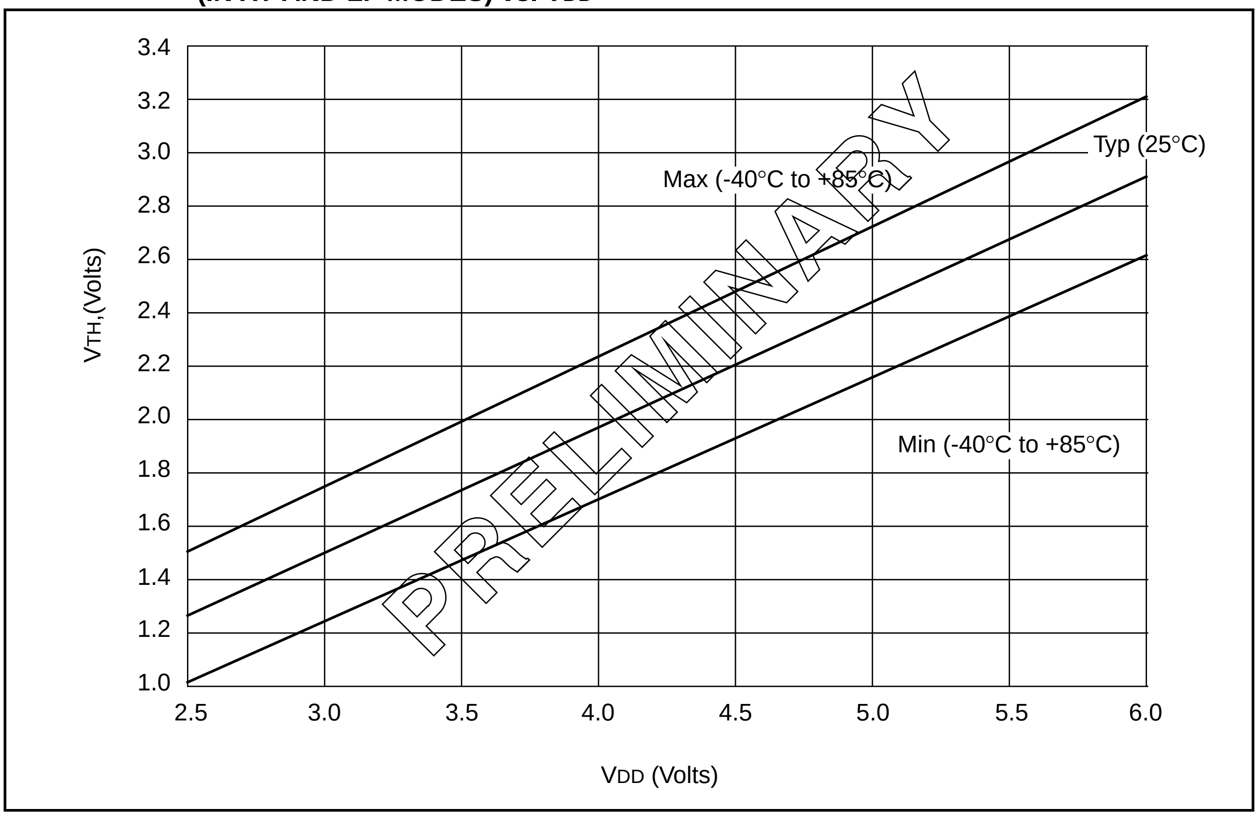
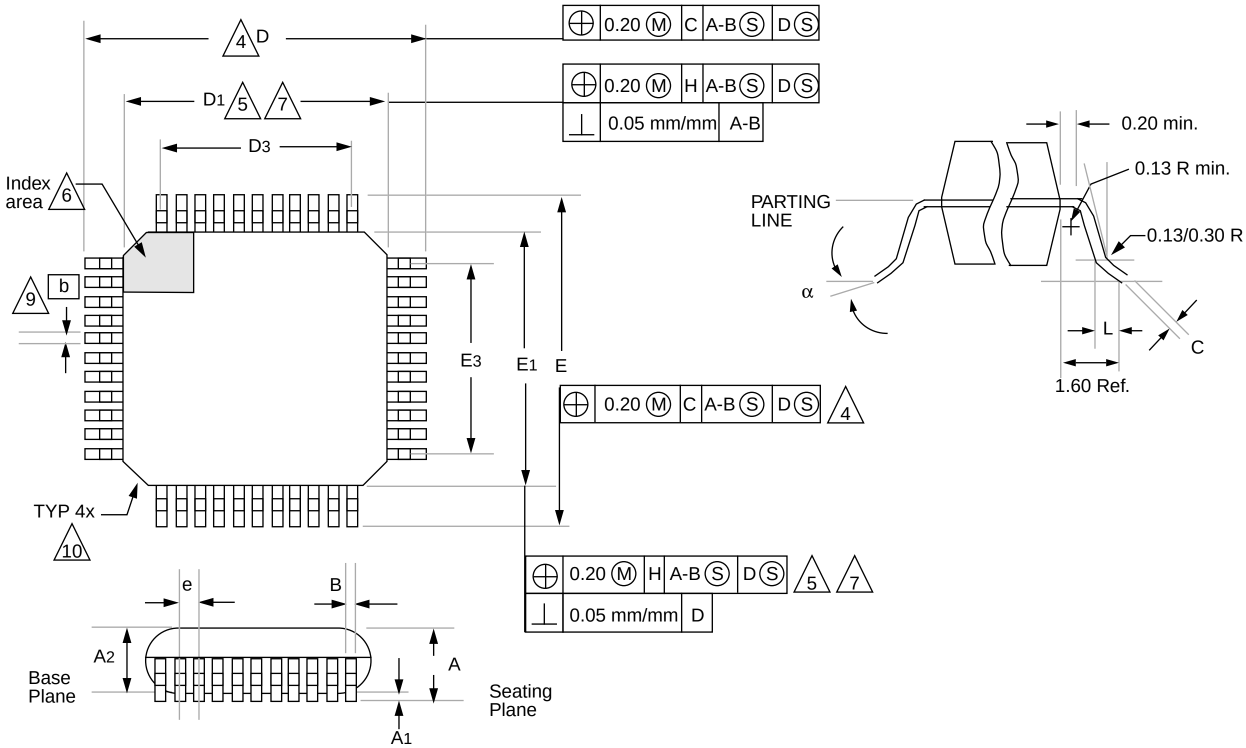


FIGURE 20-20: V_{TH} (INPUT THRESHOLD VOLTAGE) OF OSC1 INPUT (IN XT AND LF MODES) vs. V_{DD}



21.4 44-Lead Plastic Surface Mount (MQFP 10x10 mm Body 1.6/0.15 mm Lead Form)



Package Group: Plastic MQFP						
	Millimeters			Inches		
Symbol	Min	Max	Notes	Min	Max	Notes
α	0°	7°		0°	7°	
A	2.000	2.350		0.078	0.093	
A1	0.050	0.250		0.002	0.010	
A2	1.950	2.100		0.768	0.083	
b	0.300	0.450	**Typical**	0.011	0.018	**Typical**
C	0.150	0.180		0.006	0.007	
D	12.950	13.450		0.510	0.530	
D1	9.900	10.100		0.390	0.398	
D3	8.000	8.000	**Reference**	0.315	0.315	**Reference**
E	12.950	13.450		0.510	0.530	
E1	9.900	10.100		0.390	0.398	
E3	8.000	8.000	**Reference**	0.315	0.315	**Reference**
e	0.800	0.800		0.031	0.032	
L	0.730	1.030		0.028	0.041	
N	44	44		44	44	
CP	0.102	–		0.004	–	

E.7 PIC16C9XX Family Of Devices

	Clock	Memory		Peripherals						Features						
	Maximum Frequency of Operation (MHz)	EPROM Program Memory	Data Memory (bytes)	Timer Module(s)	Capture/Compare/PWM Module(s)	Serial Port(s) (SPI/I²C, USART)	Parallel Slave Port	A/D Converter (8-bit) Channels	LCD Module	Interrupt Sources	I/O Pins	Input Pins	Voltage Range (Volts)	In-Circuit Serial Programming	Brown-out Reset	Packages
PIC16C923	8	4K	176	TMR0, TMR1, TMR2	1	SPI/I²C	—	—	4 Com 32 Seg	8	25	27	3.0-6.0	Yes	—	64-pin SDIP(1), TQFP, 68-pin PLCC, DIE
PIC16C924	8	4K	176	TMR0, TMR1, TMR2	1	SPI/I²C	—	5	4 Com 32 Seg	9	25	27	3.0-6.0	Yes	—	64-pin SDIP(1), TQFP, 68-pin PLCC, DIE

All PIC16/17 Family devices have Power-on Reset, selectable Watchdog Timer, selectable code protect and high I/O current capability.
All PIC16CXX Family devices use serial programming with clock pin RB6 and data pin RB7.
Note 1: Please contact your local Microchip representative for availability of this package.

INDEX

A

ADDLW ... 112
ADDWF ... 112
ADDWFC ... 113
ALU ... 9
ALU STATUS Register (ALUSTA) ... 36
ALUSTA ... 34, 36, 108
ALUSTA Register ... 36
ANDLW ... 113
ANDWF ... 114
Application Notes
    AN552 ... 55
Assembler ... 144
Asynchronous Master Transmission ... 90
Asynchronous Transmitter ... 89

B

Bank Select Register (BSR) ... 42
Banking ... 42
Baud Rate Formula ... 86
Baud Rate Generator (BRG) ... 86
Baud Rates
    Asynchronous Mode ... 88
    Synchronous Mode ... 87
BCF ... 114
Bit Manipulation ... 108
Block Diagrams
    On-chip Reset Circuit ... 15
    PIC17C42 ... 10
    PORTD ... 60
    PORTE ... 62
    PWM ... 75
    RA0 and RA1 ... 53
    RA2 and RA3 ... 54
    RA4 and RA5 ... 54
    RB3:RB2 Port Pins ... 56
    RB7:RB4 and RB1:RB0 Port Pins ... 55
    RC7:RC0 Port Pins ... 58
    Timer3 with One Capture and One Period Register ... 78
    TMR1 and TMR2 in 16-bit Timer/Counter Mode ... 74
    TMR1 and TMR2 in Two 8-bit Timer/Counter Mode ... 73
    TMR3 with Two Capture Registers ... 79
    WDT ... 104
$\overline{\text{BORROW}}$ ... 9
BRG ... 86
Brown-out Protection ... 18
BSF ... 115
BSR ... 34, 42
BSR Operation ... 42
BTFSC ... 115
BTFSS ... 116
BTG ... 116

C

C ... 9, 36
C Compiler (MP-C) ... 145
CA1/$\overline{\text{PR3}}$ ... 72
CA1ED0 ... 71
CA1ED1 ... 71
CA1IE ... 23
CA1IF ... 24
CA1OVF ... 72
CA2ED0 ... 71
CA2ED1 ... 71
CA2H ... 20, 35
CA2IE ... 23, 78
CA2IF ... 24, 78
CA2L ... 20, 35
CA2OVF ... 72
Calculating Baud Rate Error ... 86
CALL ... 39, 117
Capacitor Selection
    Ceramic Resonators ... 101
    Crystal Oscillator ... 101
Capture ... 71, 78
Capture Sequence to Read Example ... 78
Capture1
    Mode ... 71
    Overflow ... 72
Capture2
    Mode ... 71
    Overflow ... 72
Carry (C) ... 9
Ceramic Resonators ... 100
Circular Buffer ... 39
Clearing the Prescaler ... 103
Clock/Instruction Cycle (Figure) ... 14
Clocking Scheme/Instruction Cycle (Section) ... 14
CLRF ... 117
CLRWDT ... 118
Code Protection ... 99, 106
COMF ... 118
Configuration
    Bits ... 100
    Locations ... 100
    Oscillator ... 100
    Word ... 99
CPFSEQ ... 119
CPFSGT ... 119
CPFSLT ... 120
CPU STATUS Register (CPUSTA) ... 37
CPUSTA ... 34, 37, 105
CREN ... 84
Crystal Operation, Overtone Crystals ... 101
Crystal or Ceramic Resonator Operation ... 100
Crystal Oscillator ... 100
CSRC ... 83

D

Data Memory
    GPR ... 29, 32
    Indirect Addressing ... 39
    Organization ... 32
    SFR ... 29, 32
    Transfer to Program Memory ... 43
DAW ... 120
DC ... 9, 36
DDRB ... 19, 34, 55
DDRC ... 19, 34, 58
DDRD ... 19, 34, 60
DDRE ... 19, 34, 62
DECF ... 121
DECFSNZ ... 122
DECFSZ ... 121