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	Obsolete
Core Processor	PIC
Core Size	8-Bit
Speed	16MHz
Connectivity	UART/USART
Peripherals	POR, PWM, WDT
Number of I/O	33
Program Memory Size	4KB (2K x 16)
Program Memory Type	OTP
EEPROM Size	-
RAM Size	232 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/pic17c42at-16e-pt

PIC17C4X

FIGURE 3-1: PIC17C42 BLOCK DIAGRAM

FIGURE 3-1: PIC17C42 BLOCK DIAGRAM
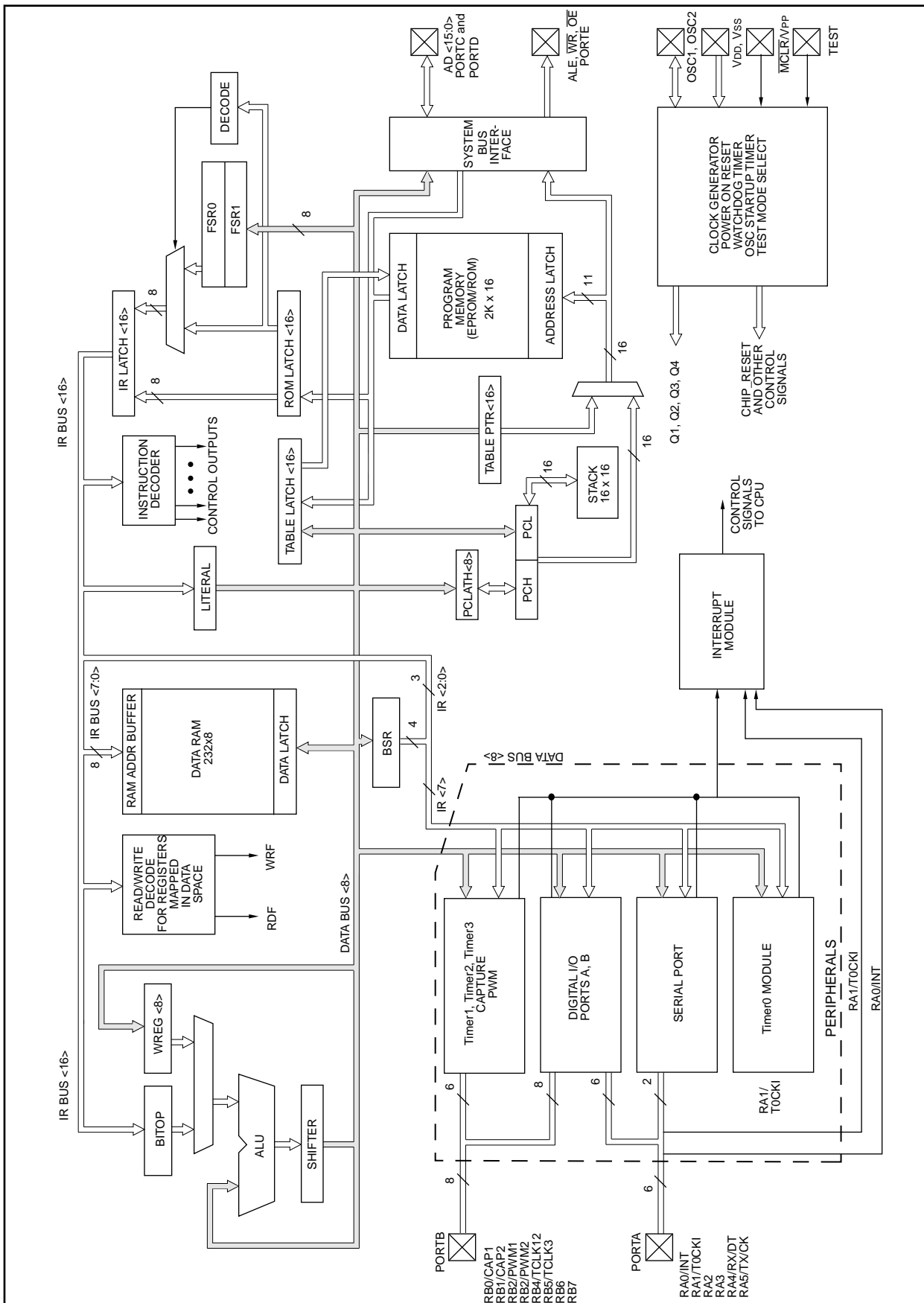
5.4 Interrupt Operation

Global Interrupt Disable bit, GLINTD (CPUSTA<4>), enables all unmasked interrupts (if clear) or disables all interrupts (if set). Individual interrupts can be disabled through their corresponding enable bits in the INTSTA register. Peripheral interrupts need either the global peripheral enable PEIE bit disabled, or the specific peripheral enable bit disabled. Disabling the peripherals via the global peripheral enable bit, disables all peripheral interrupts. GLINTD is set on reset (interrupts disabled).

The RETFIE instruction allows returning from interrupt and re-enable interrupts at the same time.

When an interrupt is responded to, the GLINTD bit is automatically set to disable any further interrupt, the return address is pushed onto the stack and the PC is loaded with interrupt vector. There are four interrupt vectors to reduce interrupt latency.

The peripheral interrupt vector has multiple interrupt sources. Once in the peripheral interrupt service routine, the source(s) of the interrupt can be determined by polling the interrupt flag bits. The peripheral interrupt flag bit(s) must be cleared in software before re-enabling interrupts to avoid continuous interrupts.

The PIC17C4X devices have four interrupt vectors. These vectors and their hardware priority are shown in Table 5-1. If two enabled interrupts occur "at the same time", the interrupt of the highest priority will be serviced first. This means that the vector address of that interrupt will be loaded into the program counter (PC).

TABLE 5-1: INTERRUPT VECTORS/PRIORITIES

Address	Vector	Priority
0008h	External Interrupt on RA0/INT pin (INTF)	1 (Highest)
0010h	TMR0 overflow interrupt (T0IF)	2
0018h	External Interrupt on T0CKI (T0CKIF)	3
0020h	Peripherals (PEIF)	4 (Lowest)

Note 1: Individual interrupt flag bits are set regardless of the status of their corresponding mask bit or the GLINTD bit.

Note 2: When disabling any of the INTSTA enable bits, the GLINTD bit should be set (disabled).

Note 3: For the PIC17C42 only:
If an interrupt occurs while the Global Interrupt Disable (GLINTD) bit is being set, the GLINTD bit may unintentionally be re-enabled by the user's Interrupt Service Routine (the RETFIE instruction). The events that would cause this to occur are:

1. An interrupt occurs simultaneously with an instruction that sets the GLINTD bit.

2. The program branches to the Interrupt vector and executes the Interrupt Service Routine.

3. The Interrupt Service Routine completes with the execution of the RETFIE instruction. This causes the GLINTD bit to be cleared (enables interrupts), and the program returns to the instruction after the one which was meant to disable interrupts.

The method to ensure that interrupts are globally disabled is:

1. Ensure that the GLINTD bit was set by the instruction, as shown in the following code:

```
LOOP    BSF     CPUSTA, GLINTD  ; Disable Global
                                ; Interrupt
        BTFSS   CPUSTA, GLINTD  ; Global Interrupt
                                ; Disabled?
        GOTO    LOOP            ; NO, try again
                                ; YES, continue
                                ; with program
                                ; low
```

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 CPUSTA 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
```

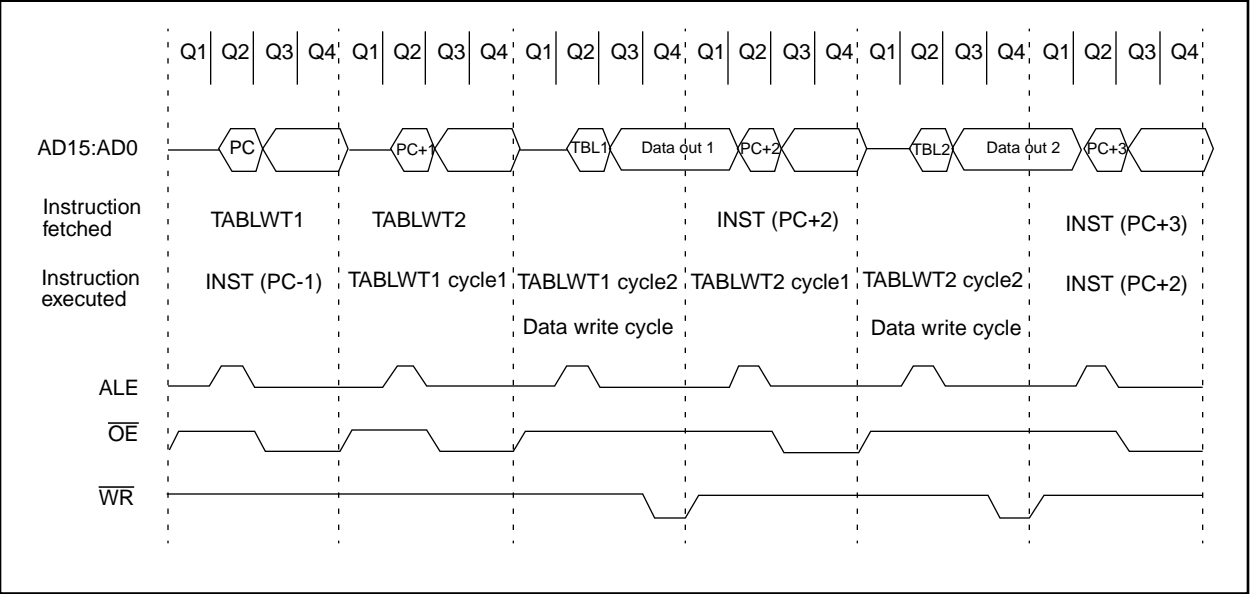**FIGURE 7-6: CONSECUTIVE TABLWT WRITE TIMING (EXTERNAL MEMORY)**

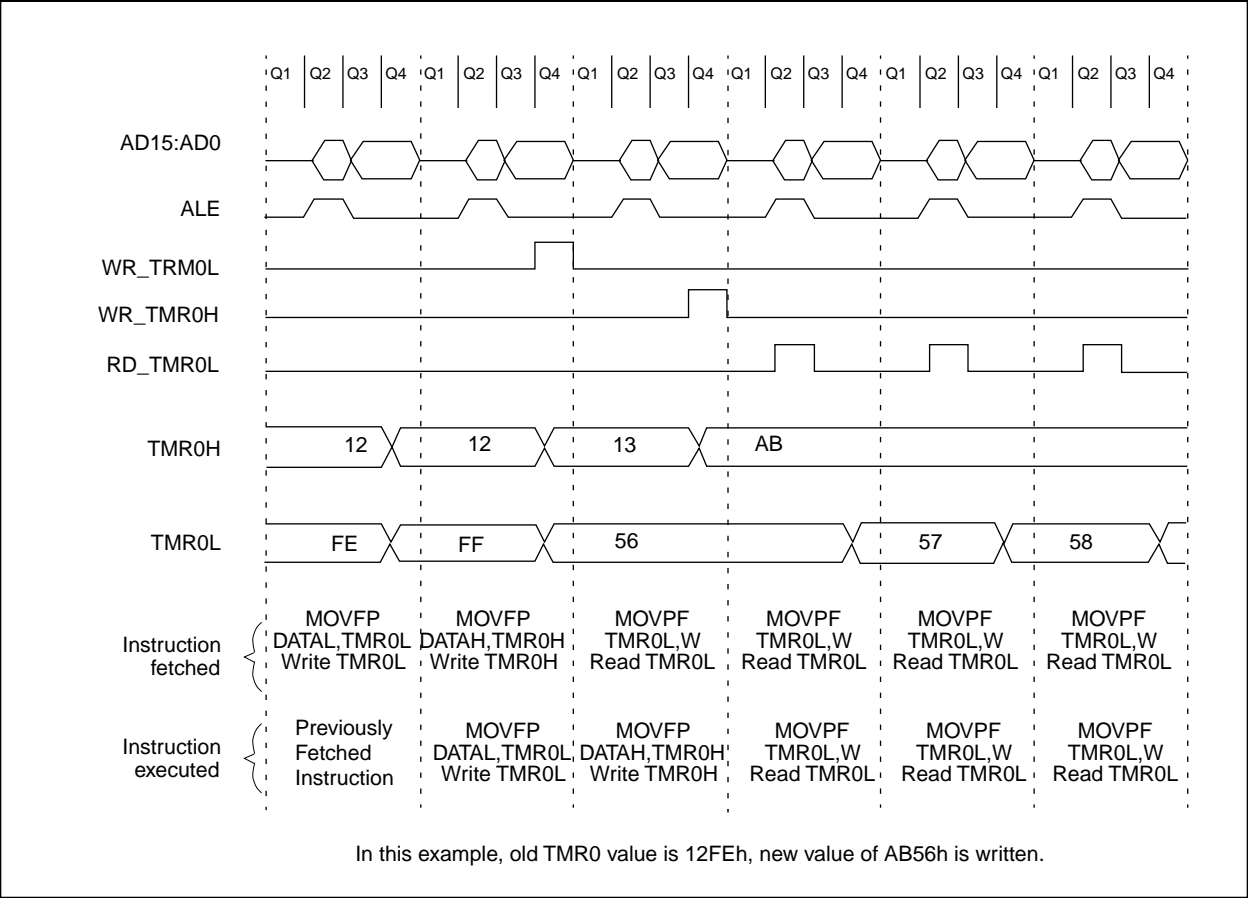## FIGURE 11-5: TMR0 READ/WRITE IN TIMER MODE

In this example, old TMR0 value is 12FEh, new value of AB56h is written.

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	$\overline{TO}$	$\overline{PD}$	—	—	--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 $\overline{MCLR}$ and the Watchdog Timer Reset.

### 13.3.2 USART SYNCHRONOUS MASTER RECEPTION

Once synchronous mode is selected, reception is enabled by setting either the SREN (RCSTA<5>) bit or the CREN (RCSTA<4>) bit. Data is sampled on the RA4/RX/DT pin on the falling edge of the clock. If SREN is set, then only a single word is received. If CREN is set, the reception is continuous until CREN is reset. If both bits are set, then CREN takes precedence. After clocking the last bit, the received data in the Receive Shift Register (RSR) is transferred to RCREG (if it is empty). If the transfer is complete, the interrupt bit RCIF (PIR<0>) is set. The actual interrupt can be enabled/disabled by setting/clearing the RCIE (PIE<0>) bit. RCIF is a read only bit which is RESET by the hardware. In this case it is reset when RCREG has been read and is empty. RCREG is a double buffered register; i.e., it is a two deep FIFO. It is possible for two bytes of data to be received and transferred to the RCREG FIFO and a third byte to begin shifting into the RSR. On the clocking of the last bit of the third byte, if RCREG is still full, then the overrun error bit OERR (RCSTA<1>) is set. The word in the RSR will be lost. RCREG can be read twice to retrieve the two bytes in the FIFO. The OERR bit has to be cleared in software. This is done by clearing the CREN bit. If OERR bit is set, transfers from RSR to RCREG are inhibited, so it is essential to clear OERR bit if it is set. The 9th receive bit is buffered the same way as the receive data. Reading the RCREG register will allow the RX9D and FERR bits to be loaded with values for the next received data; therefore, it is essential for the user to read the RCSTA register before reading RCREG in order not to lose the old FERR and RX9D information.

Steps to follow when setting up a Synchronous Master Reception:

1. Initialize the SPBRG register for the appropriate baud rate. See Section 13.1 for details.
2. Enable the synchronous master serial port by setting bits SYNC, SPEN, and CSRC.
3. If interrupts are desired, then set the RCIE bit.
4. If 9-bit reception is desired, then set the RX9 bit.
5. If a single reception is required, set bit SREN. For continuous reception set bit CREN.
6. The RCIF bit will be set when reception is complete and an interrupt will be generated if the RCIE bit was set.
7. Read RCSTA to get the ninth bit (if enabled) and determine if any error occurred during reception.
8. Read the 8-bit received data by reading RCREG.
9. If any error occurred, clear the error by clearing CREN.

Note:	To terminate a reception, either clear the SREN and CREN bits, or the SPEN bit. This will reset the receive logic, so that it will be in the proper state when receive is re-enabled.

FIGURE 13-11: SYNCHRONOUS RECEPTION (MASTER MODE, SREN)

Note: Timing diagram demonstrates SYNC master mode with SREN = 1.
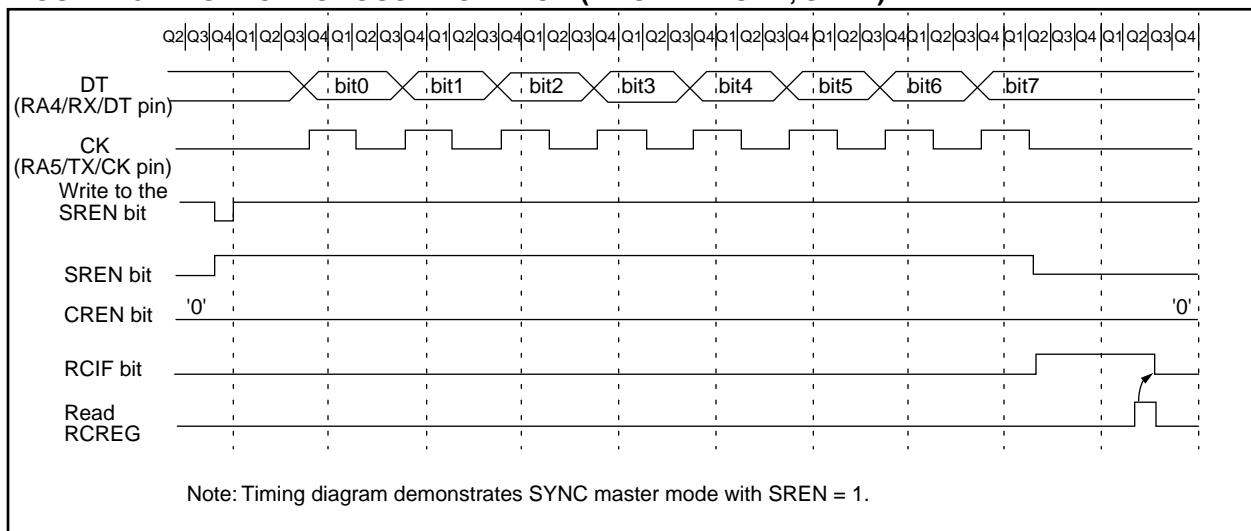
CPFSEQ	Compare f with WREG, skip if f = WREG
Syntax:	[*label*] CPFSEQ f
Operands:	0 ≤ f ≤ 255
Operation:	(f) – (WREG), skip if (f) = (WREG) (unsigned comparison)
Status Affected:	None
Encoding:	0011 0001 ffff ffff
Description:	Compares the contents of data memory location 'f' to the contents of WREG by performing an unsigned subtraction. If 'f' = 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      CPFSEQ REG
NEQUAL    :
EQUAL     :
```

Before Instruction

```
PC Address   =   HERE
WREG         =   ?
REG          =   ?
```

After Instruction

```
If REG       =   WREG;
     PC      =   Address (EQUAL)
If REG       ≠   WREG;
     PC      =   Address (NEQUAL)
```

CPFSGT	Compare f with WREG, skip if f > WREG
Syntax:	[*label*] CPFSGT f
Operands:	0 ≤ f ≤ 255
Operation:	(f) – (WREG), skip if (f) > (WREG) (unsigned comparison)
Status Affected:	None
Encoding:	0011 0010 ffff ffff
Description:	Compares the contents of data memory location 'f' to the contents of the 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       CPFSGT REG
NGREATER   :
GREATER    :
```

Before Instruction

```
PC           =   Address (HERE)
WREG         =   ?
```

After Instruction

```
If REG       >   WREG;
     PC      =   Address (GREATER)
If REG       ≤   WREG;
     PC      =   Address (NGREATER)
```

PIC17C4X

NEGW	Negate W

Syntax: [*label*]　NEGW　　f,s

Operands: 0 ≤ F ≤ 255
s ∈ [0,1]

Operation: $\overline{\text{WREG}} + 1 \rightarrow$ (f);
$\overline{\text{WREG}} + 1 \rightarrow$ s

Status Affected: OV, C, DC, Z

Encoding:

0010	110s	ffff	ffff

Description: WREG is negated using two's complement. If 's' is 0 the result is placed in WREG and data memory location 'f'. If 's' is 1 the result is placed only 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

Example: NEGW　REG,0

　Before Instruction
　　WREG　=　0011 1010 [0x3A],
　　REG　=　1010 1011 [0xAB]

　After Instruction
　　WREG　=　1100 0111 [0xC6]
　　REG　=　1100 0111 [0xC6]

NOP	No Operation

Syntax: [*label*]　NOP

Operands: None

Operation: No operation

Status Affected: None

Encoding:

0000	0000	0000	0000

Description: No operation.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	NOP	Execute	NOP

Example:

　None.

SWAPF	Swap f
Syntax:	[*label*] SWAPF f,d
Operands:	$0 \le f \le 255$ $d \in [0,1]$
Operation:	f<3:0> $\rightarrow$ dest<7:4>; f<7:4> $\rightarrow$ dest<3:0>
Status Affected:	None

Encoding:

0001	110d	ffff	ffff

Description:	The upper and lower nibbles of register 'f' are exchanged. If 'd' is 0 the result is placed in WREG. If 'd' is 1 the result is placed in register 'f'.
Words:	1
Cycles:	1

Q Cycle Activity:

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

<u>Example</u>:   SWAPF   REG,   0

   Before Instruction
      REG   =   0x53

   After Instruction
      REG   =   0x35

TABLRD	Table Read
Syntax:	[*label*] TABLRD t,i,f
Operands:	$0 \le f \le 255$ $i \in [0,1]$ $t \in [0,1]$

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

Status Affected:	None

Encoding:

1010	10ti	ffff	ffff

Description:	1. A byte of the table latch (TBLAT) is moved to register file 'f'. If t = 0: the high byte is moved; If t = 1: the low byte is moved 2. Then the contents of the program memory location pointed to by the 16-bit Table Pointer (TBLPTR) is loaded into the 16-bit Table Latch (TBLAT). 3. If i = 1: TBLPTR is incremented; If i = 0: TBLPTR is not incremented
Words:	1
Cycles:	2 (3 cycle if f = PCL)

Q Cycle Activity:

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

16.0   DEVELOPMENT SUPPORT

16.1   Development Tools

The PIC16/17 microcontrollers are supported with a full range of hardware and software development tools:

- PICMASTER/PICMASTER CE Real-Time In-Circuit Emulator
- ICEPIC Low-Cost PIC16C5X and PIC16CXXX In-Circuit Emulator
- PRO MATE® II Universal Programmer
- PICSTART® Plus Entry-Level Prototype Programmer
- PICDEM-1 Low-Cost Demonstration Board
- PICDEM-2 Low-Cost Demonstration Board
- PICDEM-3 Low-Cost Demonstration Board
- MPASM Assembler
- MPLAB-SIM Software Simulator
- MPLAB-C (C Compiler)
- Fuzzy logic development system (fuzzyTECH®–MP)

16.2   PICMASTER: High Performance Universal In-Circuit Emulator with MPLAB IDE

The PICMASTER Universal In-Circuit Emulator is intended to provide the product development engineer with a complete microcontroller design tool set for all microcontrollers in the PIC12C5XX, PIC14000, PIC16C5X, PIC16CXXX and PIC17CXX families. PICMASTER is supplied with the MPLAB™ Integrated Development Environment (IDE), which allows editing, "make" and download, and source debugging from a single environment.

Interchangeable target probes allow the system to be easily reconfigured for emulation of different processors. The universal architecture of the PICMASTER allows expansion to support all new Microchip microcontrollers.

The PICMASTER Emulator System has been designed as a real-time emulation system with advanced features that are generally found on more expensive development tools. The PC compatible 386 (and higher) machine platform and Microsoft Windows® 3.x environment were chosen to best make these features available to you, the end user.

A CE compliant version of PICMASTER is available for European Union (EU) countries.

16.3   ICEPIC: Low-cost PIC16CXXX In-Circuit Emulator

ICEPIC is a low-cost in-circuit emulator solution for the Microchip PIC16C5X and PIC16CXXX families of 8-bit OTP microcontrollers.

ICEPIC is designed to operate on PC-compatible machines ranging from 286-AT® through Pentium™ based machines under Windows 3.x environment. ICEPIC features real time, non-intrusive emulation.

16.4   PRO MATE II: Universal Programmer

The PRO MATE II Universal Programmer is a full-featured programmer capable of operating in stand-alone mode as well as PC-hosted mode.

The PRO MATE II has programmable V_{DD} and V_{PP} supplies which allows it to verify programmed memory at V_{DD} min and V_{DD} max for maximum reliability. It has an LCD display for displaying error messages, keys to enter commands and a modular detachable socket assembly to support various package types. In stand-alone mode the PRO MATE II can read, verify or program PIC16C5X, PIC16CXXX, PIC17CXX and PIC14000 devices. It can also set configuration and code-protect bits in this mode.

16.5   PICSTART Plus Entry Level Development System

The PICSTART programmer is an easy-to-use, low-cost prototype programmer. It connects to the PC via one of the COM (RS-232) ports. MPLAB Integrated Development Environment software makes using the programmer simple and efficient. PICSTART Plus is not recommended for production programming.

PICSTART Plus supports all PIC12C5XX, PIC14000, PIC16C5X, PIC16CXXX and PIC17CXX devices with up to 40 pins. Larger pin count devices such as the PIC16C923 and PIC16C924 may be supported with an adapter socket.

17.3 Timing Parameter Symbology

The timing parameter symbols have been created using one of the following formats:

1. TppS2ppS
2. TppS

T				
F	Frequency	T	Time	

Lowercase symbols (pp) and their meanings:

pp				
ad	Address/Data	ost	Oscillator Start-up Timer	
al	ALE	pwrt	Power-up Timer	
cc	Capture1 and Capture2	rb	PORTB	
ck	CLKOUT or clock	rd	$\overline{RD}$	
dt	Data in	rw	$\overline{RD}$ or $\overline{WR}$	
in	INT pin	t0	T0CKI	
io	I/O port	t123	TCLK12 and TCLK3	
mc	$\overline{MCLR}$	wdt	Watchdog Timer	
oe	$\overline{OE}$	wr	$\overline{WR}$	
os	OSC1			

Uppercase symbols and their meanings:

S				
D	Driven	L	Low	
E	Edge	P	Period	
F	Fall	R	Rise	
H	High	V	Valid	
I	Invalid (Hi-impedance)	Z	Hi-impedance	

FIGURE 17-11: MEMORY INTERFACE WRITE TIMING
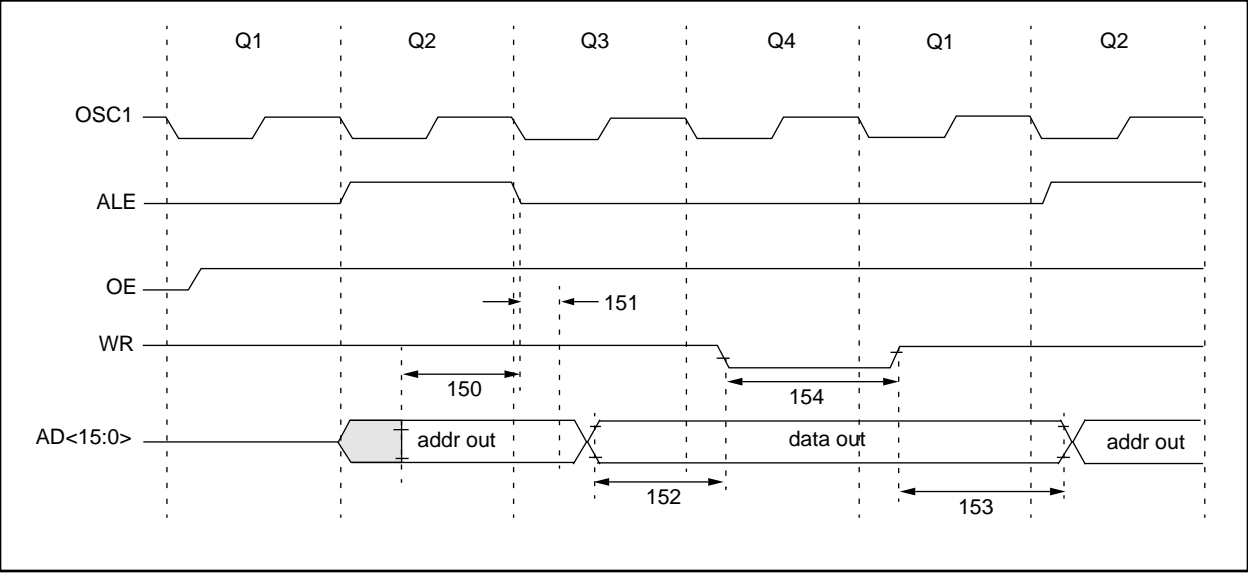
TABLE 17-11: MEMORY INTERFACE WRITE REQUIREMENTS

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
150	TadV2alL	AD<15:0> (address) valid to ALE↓ (address setup time)	0.25Tcy - 30	—	—	ns	
151	TalL2adl	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	TwrH2adl	$\overline{WR}$↑ to data out invalid (data hold time)	—	0.25T$_{CY}$ §	—	ns	
154	TwrL	$\overline{WR}$ pulse width	—	0.25T$_{CY}$ §	—	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 is guaranteed by design.

PIC17C4X

FIGURE 18-2: TYPICAL RC OSCILLATOR FREQUENCY vs. V$_{DD}$
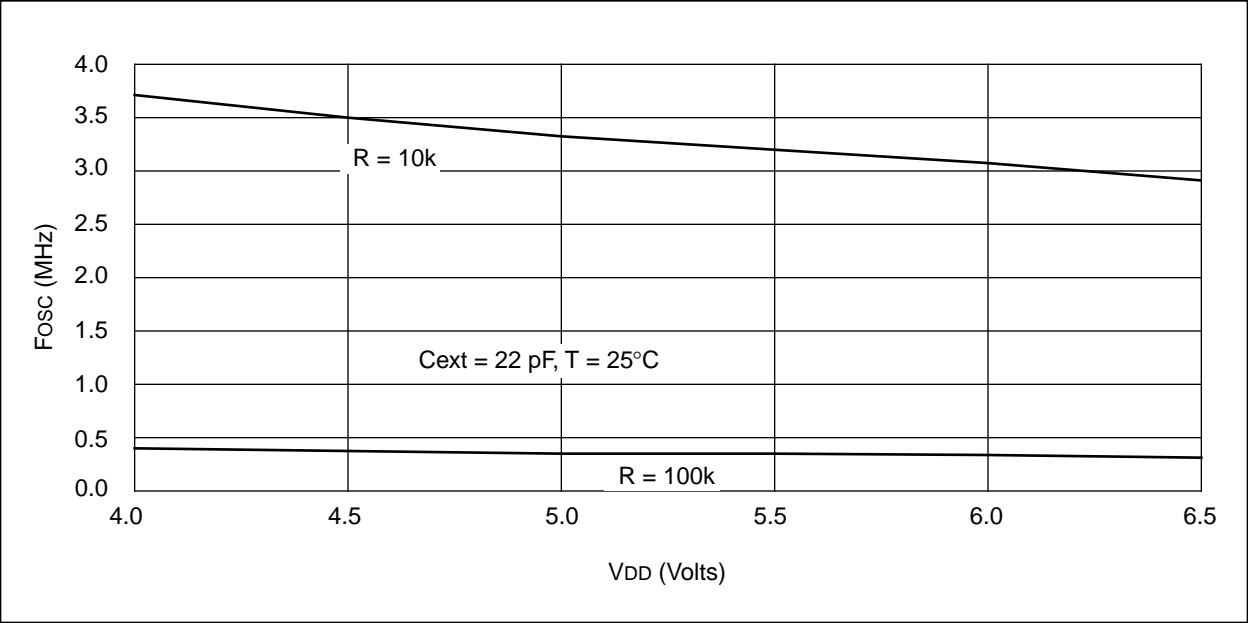
FIGURE 18-3: TYPICAL RC OSCILLATOR FREQUENCY vs. V$_{DD}$
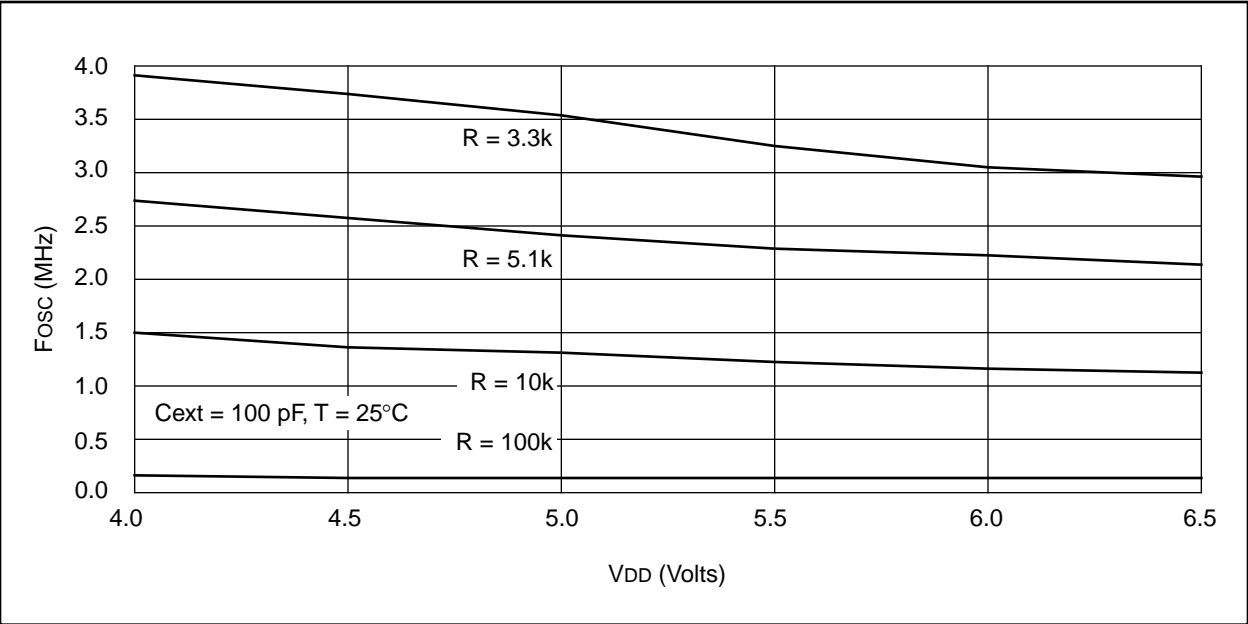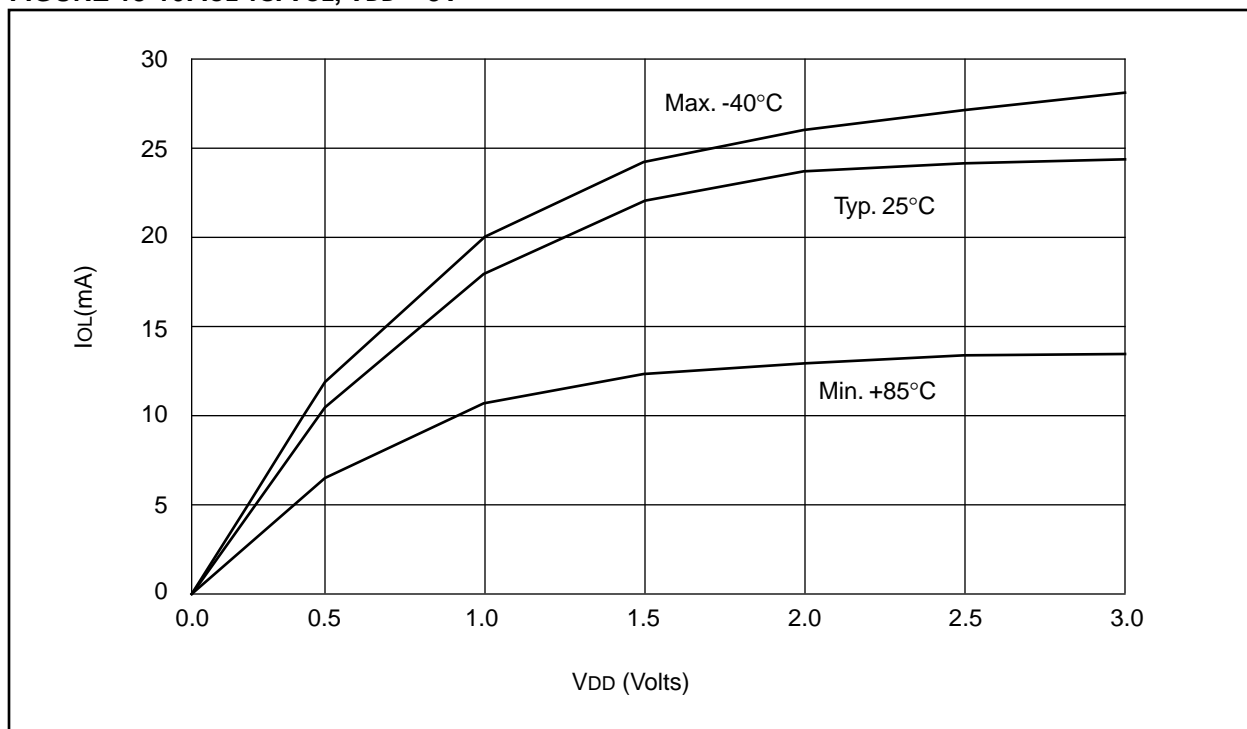
PIC17C4X

FIGURE 18-15: IOH vs. VOH, VDD = 5V
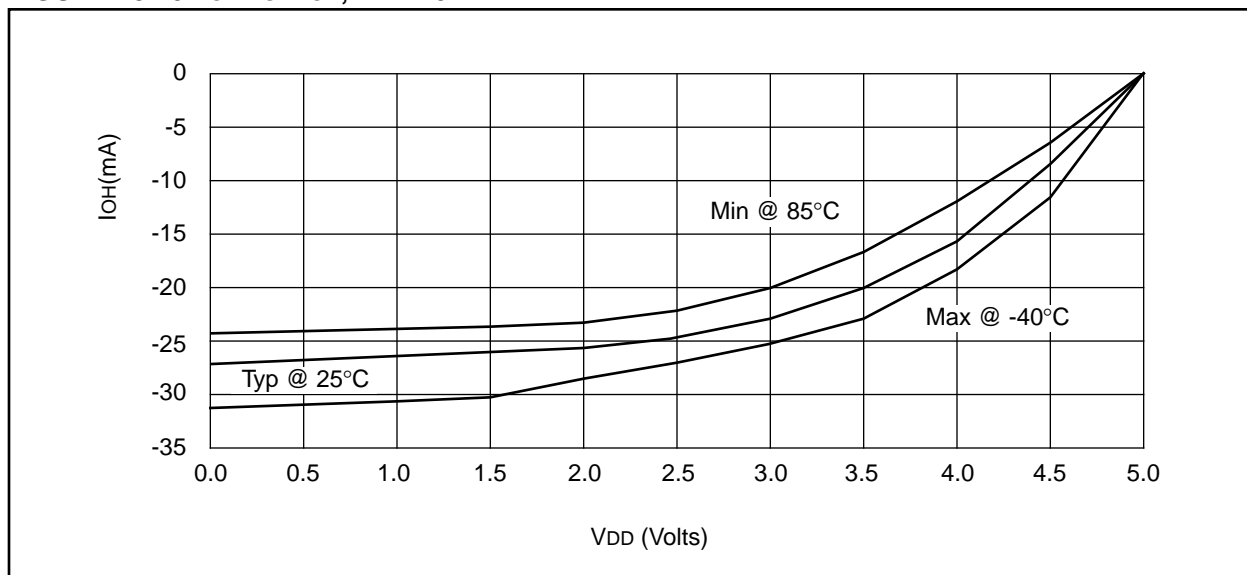
FIGURE 18-16: IOL vs. VOL, VDD = 3V

19.0 PIC17CR42/42A/43/R43/44 ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings †

Ambient temperature under bias...-55 to +125˚C

Storage temperature .... -65˚C to +150˚C

Voltage on V_{DD} with respect to V_{SS} .... 0 to +7.5V

Voltage on $\overline{MCLR}$ with respect to V_{SS} (Note 2) ....-0.6V to +14V

Voltage on RA2 and RA3 with respect to V_{SS}...-0.6V to +14V

Voltage on all other pins with respect to V_{SS} .... -0.6V to V_{DD} + 0.6V

Total power dissipation (Note 1)..1.0W

Maximum current out of V_{SS} pin(s) - total ....250 mA

Maximum current into V_{DD} pin(s) - total..200 mA

Input clamp current, I_{IK} (V_I < 0 or V_I > V_{DD}) ....±20 mA

Output clamp current, I_{OK} (V_O < 0 or V_O > V_{DD})..±20 mA

Maximum output current sunk by any I/O pin (except RA2 and RA3).......................................35 mA

Maximum output current sunk by RA2 or RA3 pins ....60 mA

Maximum output current sourced by any I/O pin ....20 mA

Maximum current sunk by PORTA and PORTB (combined)..150 mA

Maximum current sourced by PORTA and PORTB (combined)..100 mA

Maximum current sunk by PORTC, PORTD and PORTE (combined)...................................150 mA

Maximum current sourced by PORTC, PORTD and PORTE (combined)..............................100 mA

Note 1: Power dissipation is calculated as follows: Pdis = V_{DD} x {I_{DD} - Σ I_{OH}} + Σ {(V_{DD}-V_{OH}) x I_{OH}} + Σ(V_{OL} x I_{OL})

Note 2: Voltage spikes below V_{SS} at the $\overline{MCLR}$ pin, inducing currents greater than 80 mA, may cause latch-up. Thus, a series resistor of 50-100Ω should be used when applying a "low" level to the $\overline{MCLR}$ pin rather than pulling this pin directly to V_{SS}.

† NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

PIC17C4X

19.4   Timing Parameter Symbology

The timing parameter symbols have been created following one of the following formats:

1. TppS2ppS
2. TppS

3. T$_{CC:ST}$ (I^{2}C specifications only)
4. Ts   (I^{2}C specifications only)

T			
F	Frequency	T	Time

Lowercase symbols (pp) and their meanings:

pp			
ad	Address/Data	ost	Oscillator Start-Up Timer
al	ALE	pwrt	Power-Up Timer
cc	Capture1 and Capture2	rb	PORTB
ck	CLKOUT or clock	rd	$\overline{RD}$
dt	Data in	rw	$\overline{RD}$ or $\overline{WR}$
in	INT pin	t0	T0CKI
io	I/O port	t123	TCLK12 and TCLK3
mc	$\overline{MCLR}$	wdt	Watchdog Timer
oe	$\overline{OE}$	wr	$\overline{WR}$
os	OSC1		

Uppercase symbols and their meanings:

S			
D	Driven	L	Low
E	Edge	P	Period
F	Fall	R	Rise
H	High	V	Valid
I	Invalid (Hi-impedance)	Z	Hi-impedance

FIGURE 20-11: TYPICAL IPD vs. VDD WATCHDOG ENABLED 25°C
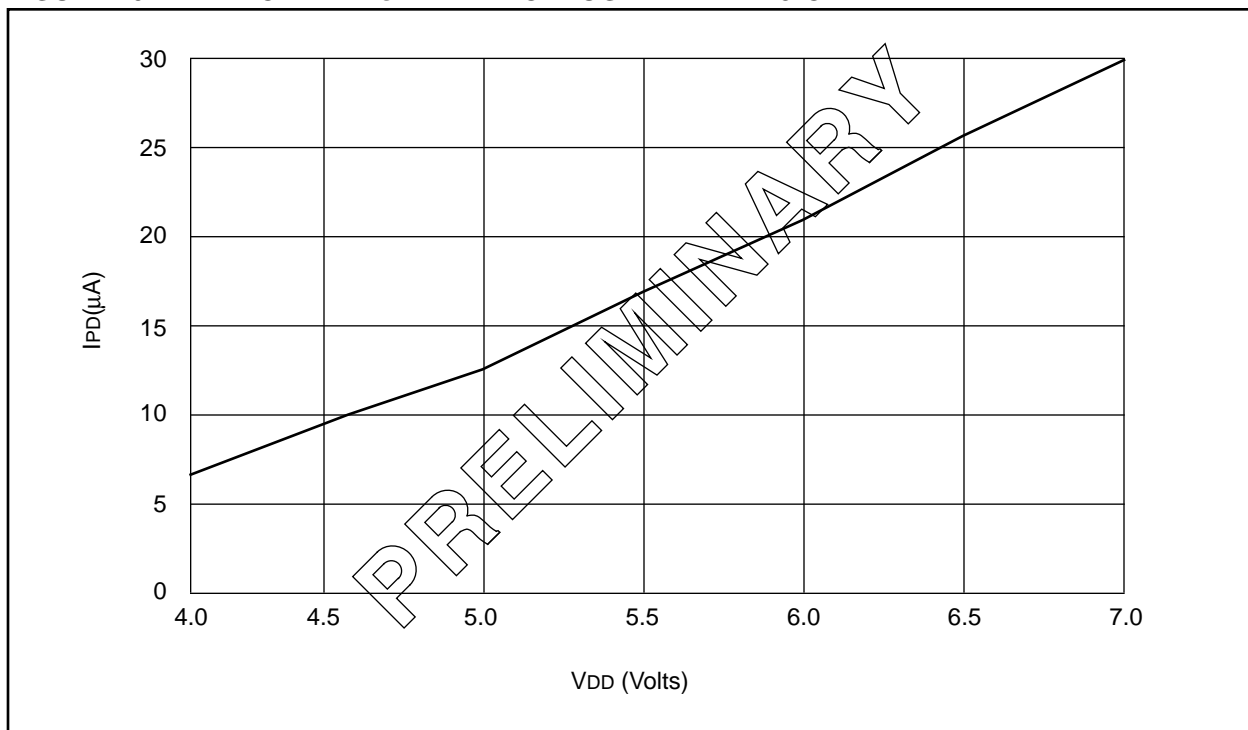
FIGURE 20-12: MAXIMUM IPD vs. VDD WATCHDOG ENABLED

NOTES:

PIC17C4X

LIST OF EXAMPLES

LIST OF FIGURES

PIC17C4X

READER RESPONSE

It is our intention to provide you with the best documentation possible to ensure successful use of your Microchip product. If you wish to provide your comments on organization, clarity, subject matter, and ways in which our documentation can better serve you, please FAX your comments to the Technical Publications Manager at (602) 786-7578.

Please list the following information, and use this outline to provide us with your comments about this Data Sheet.

To: Technical Publications Manager   Total Pages Sent
RE: Reader Response

From: Name _____

Company _____

Address _____

City / State / ZIP / Country _____

Telephone: (_____) _____ - _____   FAX: (_____) _____ - _____

Application (optional):

Would you like a reply?____Y ____N

Device: **PIC17C4X**   Literature Number: **DS30412C**

Questions:

1.   What are the best features of this document?

_____

_____

2.   How does this document meet your hardware and software development needs?

_____

_____

3.   Do you find the organization of this data sheet easy to follow? If not, why?

_____

_____

4.   What additions to the data sheet do you think would enhance the structure and subject?

_____

_____

5.   What deletions from the data sheet could be made without affecting the overall usefulness?

_____

_____

6.   Is there any incorrect or misleading information (what and where)?

_____

_____

7.   How would you improve this document?

_____

_____

8.   How would you improve our software, systems, and silicon products?

_____

_____