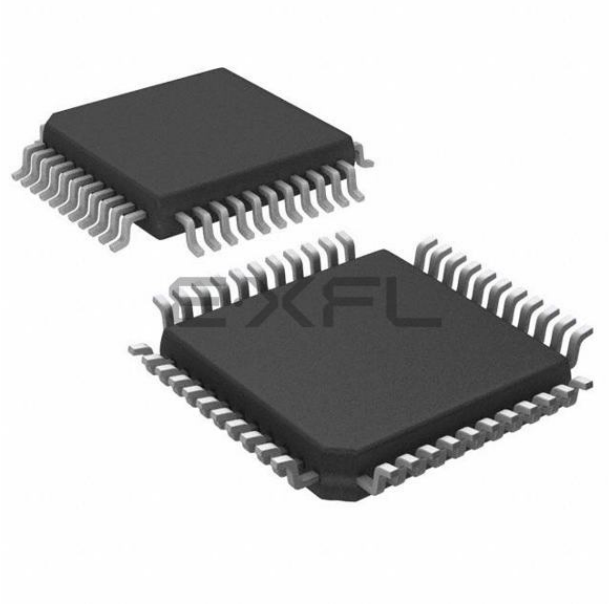


Welcome to E-XFL.COM

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	8KB (4K 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-QFP
Supplier Device Package	44-MQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic17c43-16i-pq

3.0 ARCHITECTURAL OVERVIEW

The high performance of the PIC17C4X can be attributed to a number of architectural features commonly found in RISC microprocessors. To begin with, the PIC17C4X uses a modified Harvard architecture. This architecture has the program and data accessed from separate memories. So the device has a program memory bus and a data memory bus. This improves bandwidth over traditional von Neumann architecture, where program and data are fetched from the same memory (accesses over the same bus). Separating program and data memory further allows instructions to be sized differently than the 8-bit wide data word. PIC17C4X opcodes are 16-bits wide, enabling single word instructions. The full 16-bit wide program memory bus fetches a 16-bit instruction in a single cycle. A two-stage pipeline overlaps fetch and execution of instructions. Consequently, all instructions execute in a single cycle (121 ns @ 33 MHz), except for program branches and two special instructions that transfer data between program and data memory.

The PIC17C4X can address up to 64K x 16 of program memory space.

The **PIC17C42** and **PIC17C42A** integrate 2K x 16 of EPROM program memory on-chip, while the **PIC17CR42** has 2K x 16 of ROM program memory on-chip.

The **PIC17C43** integrates 4K x 16 of EPROM program memory, while the **PIC17CR43** has 4K x 16 of ROM program memory.

The **PIC17C44** integrates 8K x 16 EPROM program memory.

Program execution can be internal only (microcontroller or protected microcontroller mode), external only (microprocessor mode) or both (extended microcontroller mode). Extended microcontroller mode does not allow code protection.

The PIC17CXX can directly or indirectly address its register files or data memory. All special function registers, including the Program Counter (PC) and Working Register (WREG), are mapped in the data memory. The PIC17CXX has an orthogonal (symmetrical) instruction set that makes it possible to carry out any operation on any register using any addressing mode. This symmetrical nature and lack of 'special optimal situations' make programming with the PIC17CXX simple yet efficient. In addition, the learning curve is reduced significantly.

One of the PIC17CXX family architectural enhancements from the PIC16CXX family allows two file registers to be used in some two operand instructions. This allows data to be moved directly between two registers without going through the WREG register. This increases performance and decreases program memory usage.

The PIC17CXX devices contain an 8-bit ALU and working register. The ALU is a general purpose arithmetic unit. It performs arithmetic and Boolean functions between data in the working register and any register file.

The ALU is 8-bits wide and capable of addition, subtraction, shift, and logical operations. Unless otherwise mentioned, arithmetic operations are two's complement in nature.

The WREG register is an 8-bit working register used for ALU operations.

All PIC17C4X devices (except the PIC17C42) have an 8 x 8 hardware multiplier. This multiplier generates a 16-bit result in a single cycle.

Depending on the instruction executed, the ALU may affect the values of the Carry (C), Digit Carry (DC), and Zero (Z) bits in the STATUS register. The C and DC bits operate as a $\overline{\text{borrow}}$ and $\overline{\text{digit borrow}}$ out bit, respectively, in subtraction. See the `SUBLW` and `SUBWF` instructions for examples.

Although the ALU does not perform signed arithmetic, the Overflow bit (OV) can be used to implement signed math. Signed arithmetic is comprised of a magnitude and a sign bit. The overflow bit indicates if the magnitude overflows and causes the sign bit to change state. Signed math can have greater than 7-bit values (magnitude), if more than one byte is used. The use of the overflow bit only operates on bit6 (MSb of magnitude) and bit7 (sign bit) of the value in the ALU. That is, the overflow bit is not useful if trying to implement signed math where the magnitude, for example, is 11-bits. If the signed math values are greater than 7-bits (15-, 24- or 31-bit), the algorithm must ensure that the low order bytes ignore the overflow status bit.

Care should be taken when adding and subtracting signed numbers to ensure that the correct operation is executed. Example 3-1 shows an item that must be taken into account when doing signed arithmetic on an ALU which operates as an unsigned machine.

EXAMPLE 3-1: SIGNED MATH

```
Hex Value       Signed Value     Unsigned Value
                Math             Math

  FFh             -127             255
+ 01h           +    1           +   1
=  ?            = -126 (FEh)     =   0 (00h);
                                 Carry bit = 1
```

`Signed math requires the result in REG to be FEh (-126). This would be accomplished by subtracting one as opposed to adding one.`

Simplified block diagrams are shown in Figure 3-1 and Figure 3-2. The descriptions of the device pins are listed in Table 3-1.

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

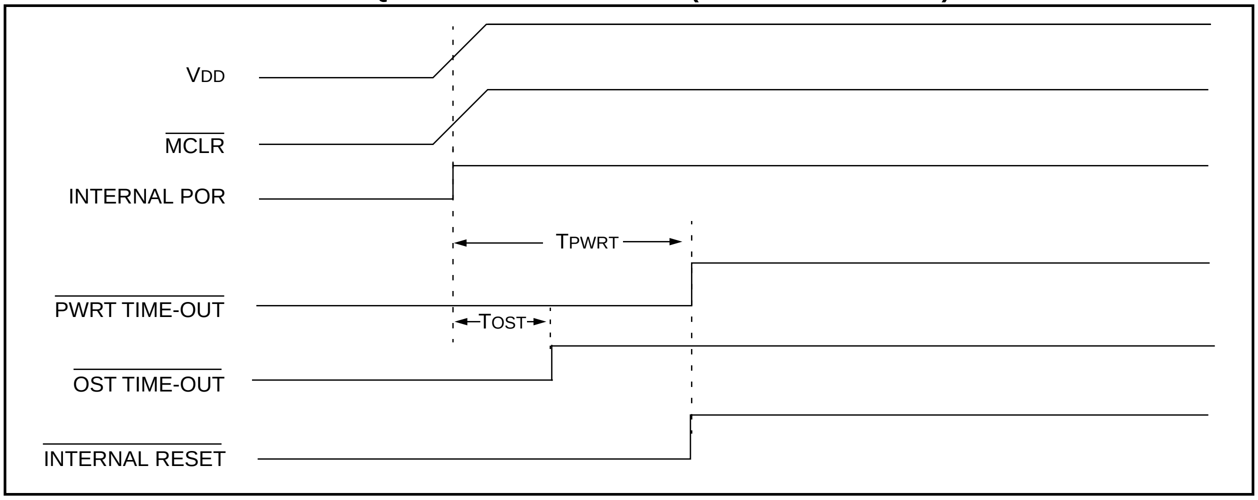


FIGURE 4-3: TIME-OUT SEQUENCE ON POWER-UP ($\overline{MCLR}$ NOT TIED TO VDD)

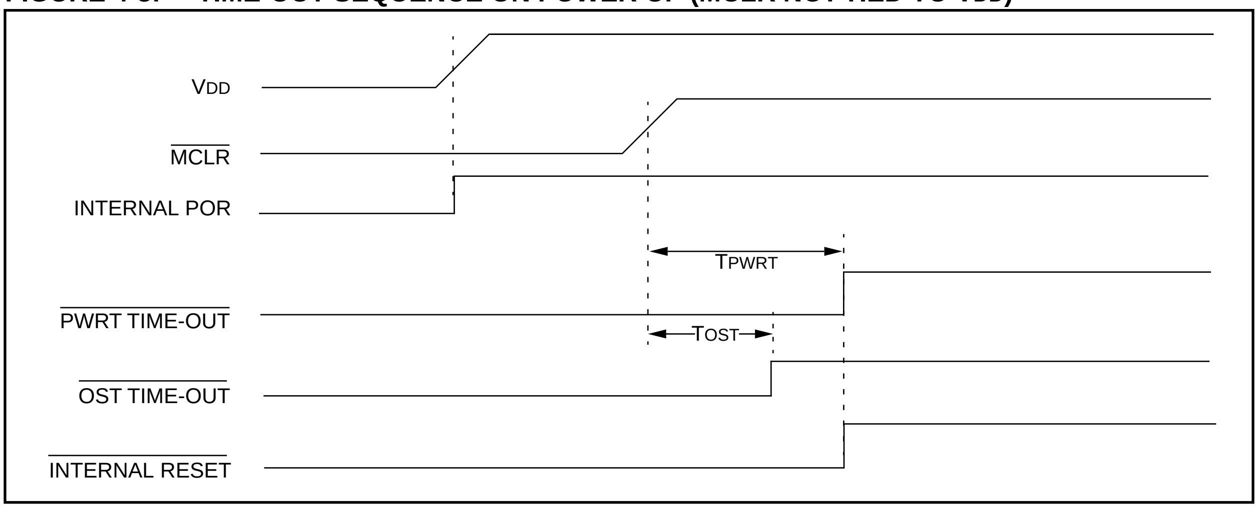
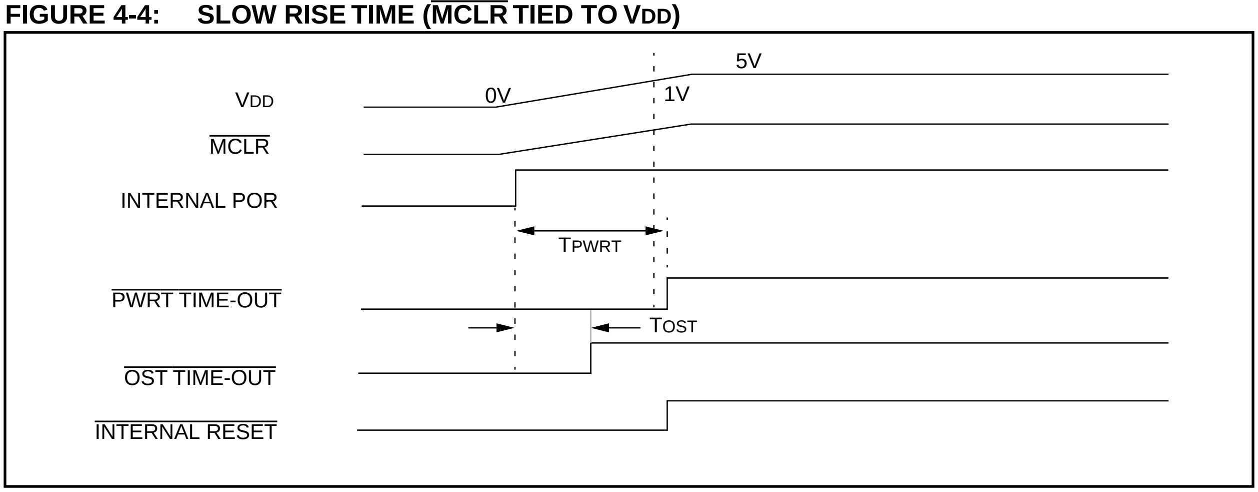


FIGURE 4-4: SLOW RISE TIME ($\overline{MCLR}$ TIED TO VDD)



6.2.2.2 CPU STATUS REGISTER (CPUSTA)

The CPUSTA register contains the status and control bits for the CPU. This register is used to globally enable/disable interrupts. If only a specific interrupt is desired to be enabled/disabled, please refer to the INTerrupt STAtus (INTSTA) register and the Peripheral Interrupt Enable (PIE) register. This register also indicates if the stack is available and contains the Power-down ($\overline{PD}$) and Time-out ($\overline{TO}$) bits. The $\overline{TO}$, $\overline{PD}$, and STKAV bits are not writable. These bits are set and cleared according to device logic. Therefore, the result of an instruction with the CPUSTA register as destination may be different than intended.

FIGURE 6-8: CPUSTA REGISTER (ADDRESS: 06h, UNBANKED)

U - 0	U - 0	R - 1	R/W - 1	R - 1	R - 1	U - 0	U - 0
—	—	STKAV	GLINTD	$\overline{TO}$	$\overline{PD}$	—	—
bit7							bit0

R = Readable bit
W = Writable bit
U = Unimplemented bit, Read as ‘0’
\- n = Value at POR reset

bit 7-6: **Unimplemented**: Read as '0'

bit 5: **STKAV**: Stack Available bit
This bit indicates that the 4-bit stack pointer value is Fh, or has rolled over from Fh → 0h (stack overflow).
1 = Stack is available
0 = Stack is full, or a stack overflow may have occurred (Once this bit has been cleared by a stack overflow, only a device reset will set this bit)

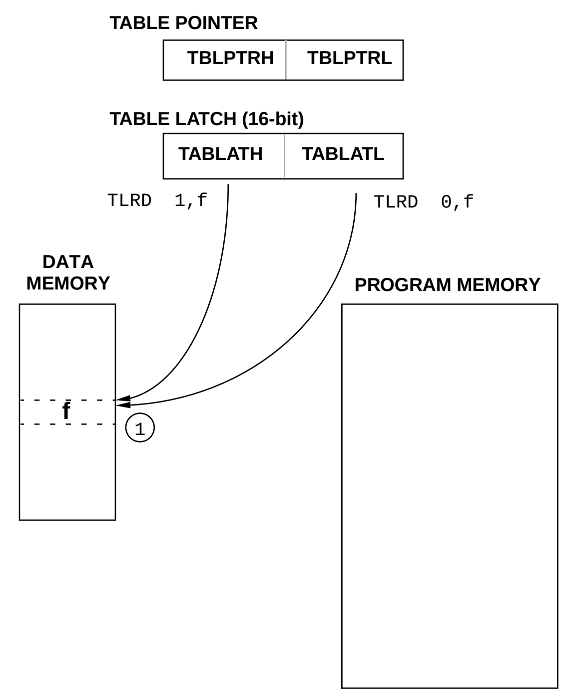
bit 4: **GLINTD**: Global Interrupt Disable bit
This bit disables all interrupts. When enabling interrupts, only the sources with their enable bits set can cause an interrupt.
1 = Disable all interrupts
0 = Enables all un-masked interrupts

bit 3: **$\overline{TO}$**: WDT Time-out Status bit
1 = After power-up or by a `CLRWDT` instruction
0 = A Watchdog Timer time-out occurred

bit 2: **$\overline{PD}$**: Power-down Status bit
1 = After power-up or by the `CLRWDT` instruction
0 = By execution of the `SLEEP` instruction

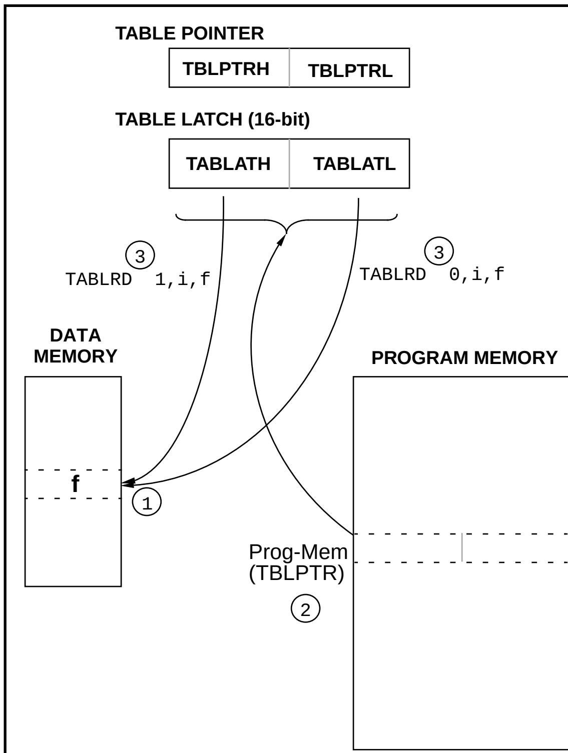
bit 1-0: **Unimplemented**: Read as '0'

FIGURE 7-3: TLRD INSTRUCTION OPERATION



Note 1: 8-bit value, from TABLAT (16-bit) high or low byte, loaded into register 'f'.

FIGURE 7-4: TABLRD INSTRUCTION OPERATION



Note 1: 8-bit value, from TABLAT (16-bit) high or low byte, loaded into register 'f'.
2: 16-bit value at Program Memory (TBLPTR) loaded into TABLAT register.
3: If “i” = 1, then TBLPTR = TBLPTR + 1, If “i” = 0, then TBLPTR is unchanged.

7.2 Table Writes to External Memory

Table writes to external memory are always two-cycle instructions. The second cycle writes the data to the external memory location. The sequence of events for an external memory write are the same for an internal write.

> **Note:** If an interrupt is pending or occurs during the TABLWT, the two cycle table write completes. The RA0/INT, TMR0, or T0CKI interrupt flag is automatically cleared or the pending peripheral interrupt is acknowledged.

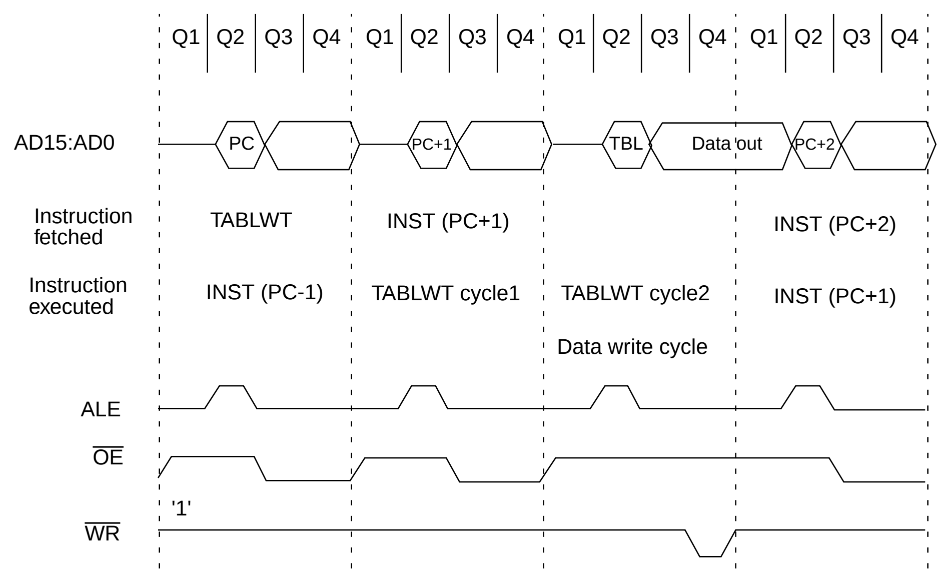
7.2.2 TABLE WRITE CODE

The "i" operand of the TABLWT instruction can specify that the value in the 16-bit TBLPTR register is automatically incremented for the next write. In Example 7-1, the TBLPTR register is not automatically incremented.

EXAMPLE 7-1: TABLE WRITE

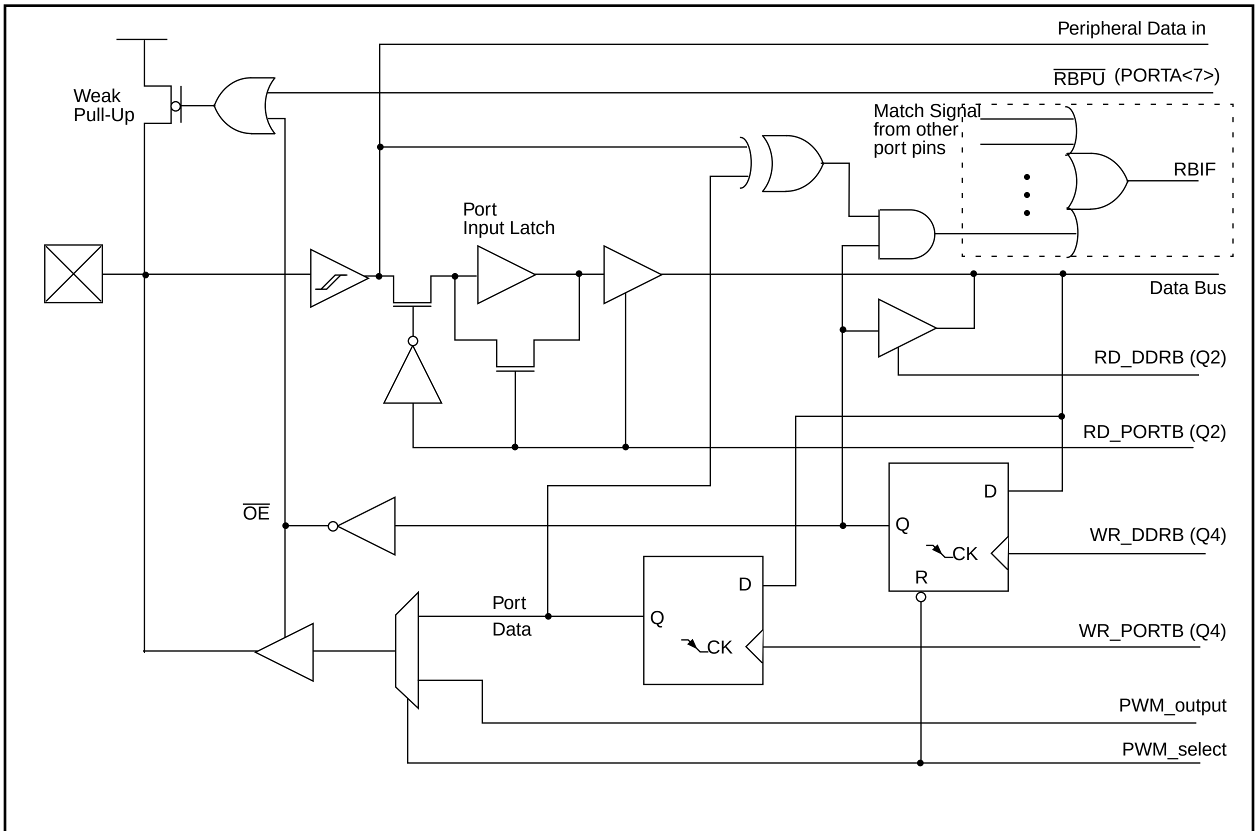
```
CLRWDT                      ; Clear WDT
MOVLW   HIGH (TBL_ADDR)     ; Load the Table
MOVWF   TBLPTRH             ;   address
MOVLW   LOW (TBL_ADDR)      ;
MOVWF   TBLPTRL             ;
MOVLW   HIGH (DATA)         ; Load HI byte
TLWT    1, WREG             ;   in TABLATCH
MOVLW   LOW (DATA)          ; Load LO byte
TABLWT  0,0,WREG            ;   in TABLATCH
                            ;   and write to
                            ;   program memory
                            ;   (Ext. SRAM)
```

FIGURE 7-5: TABLWT WRITE TIMING (EXTERNAL MEMORY)



Note: If external write GLINTD = '1', Enable bit = '1', '1' → Flag bit, Do table write. The highest pending interrupt is cleared.

FIGURE 9-5: BLOCK DIAGRAM OF RB3 AND RB2 PORT PINS



Note: I/O pins have protection diodes to VDD and Vss.

12.2.3 EXTERNAL CLOCK INPUT FOR TIMER3

When TMR3CS is set, the 16-bit TMR3 increments on the falling edge of clock input TCLK3. The input on the RB5/TCLK3 pin is sampled and synchronized by the internal phase clocks twice every instruction cycle. This causes a delay from the time a falling edge appears on TCLK3 to the time TMR3 is actually incremented. For the external clock input timing requirements, see the Electrical Specification section. Figure 12-9 shows the timing diagram when operating from an external clock.

12.2.4 READING/WRITING TIMER3

Since Timer3 is a 16-bit timer and only 8-bits at a time can be read or written, care should be taken when reading or writing while the timer is running. The best method to read or write the timer is to stop the timer, perform any read or write operation, and then restart Timer3 (using the TMR3ON bit). However, if it is necessary to keep Timer3 free-running, care must be taken. For writing to the 16-bit TMR3, Example 12-2 may be used. For reading the 16-bit TMR3, Example 12-3 may be used. Interrupts must be disabled during this routine.

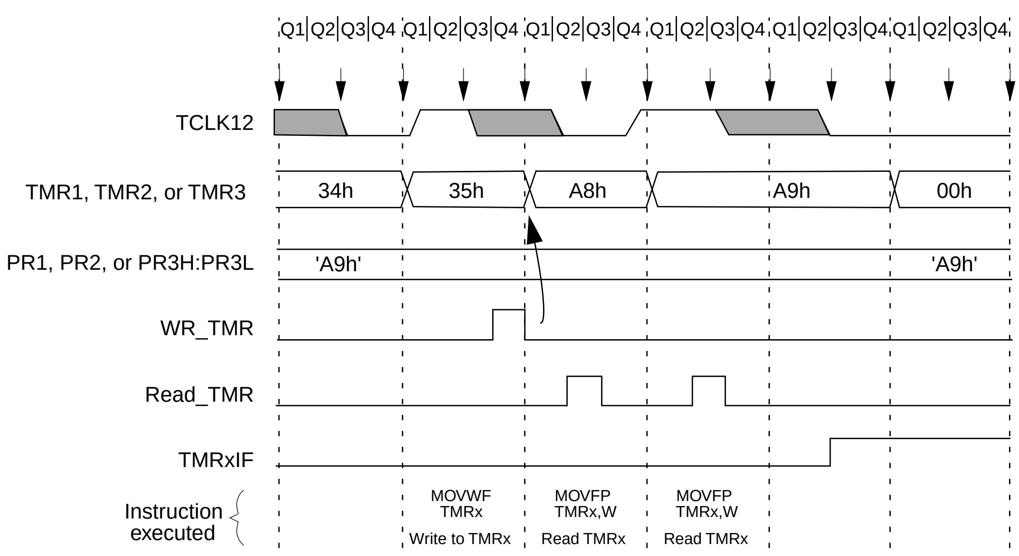
EXAMPLE 12-2: WRITING TO TMR3

```
BSF   CPUSTA, GLINTD ;Disable interrupt
MOVFP RAM_L,  TMR3L  ;
MOVFP RAM_H,  TMR3H  ;
BCF   CPUSTA, GLINTD ;Done,enable interrupt
```

EXAMPLE 12-3: READING FROM TMR3

```
MOVPF   TMR3L, TMPLO    ;read low tmr0
MOVPF   TMR3H, TMPHI    ;read high tmr0
MOVFP   TMPLO, WREG     ;tmplo —> wreg
CPFSLT  TMR3L, WREG     ;tmr0l < wreg?
RETURN                  ;no then return
MOVPF   TMR3L, TMPLO    ;read low tmr0
MOVPF   TMR3H, TMPHI    ;read high tmr0
RETURN                  ;return
```

FIGURE 12-9: TMR1, TMR2, AND TMR3 OPERATION IN EXTERNAL CLOCK MODE



Note 1: TCLK12 is sampled in Q2 and Q4.
2: ↓ indicates a sampling point.
3: The latency from TCLK12 ↓ to timer increment is between 2Tosc and 6Tosc.

TABLE 15-2: PIC17CXX INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	16-bit Opcode MSb LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS						
ADDWF	**f,d**	ADD WREG to f	1	`0000 111d ffff ffff`	OV,C,DC,Z	
ADDWFC	**f,d**	ADD WREG and Carry bit to f	1	`0001 000d ffff ffff`	OV,C,DC,Z	
ANDWF	**f,d**	AND WREG with f	1	`0000 101d ffff ffff`	Z	
CLRF	**f,s**	Clear f, or Clear f and Clear WREG	1	`0010 100s ffff ffff`	None	3
COMF	**f,d**	Complement f	1	`0001 001d ffff ffff`	Z	
CPFSEQ	**f**	Compare f with WREG, skip if f = WREG	1 (2)	`0011 0001 ffff ffff`	None	6,8
CPFSGT	**f**	Compare f with WREG, skip if f > WREG	1 (2)	`0011 0010 ffff ffff`	None	2,6,8
CPFSLT	**f**	Compare f with WREG, skip if f < WREG	1 (2)	`0011 0000 ffff ffff`	None	2,6,8
DAW	**f,s**	Decimal Adjust WREG Register	1	`0010 111s ffff ffff`	C	3
DECF	**f,d**	Decrement f	1	`0000 011d ffff ffff`	OV,C,DC,Z	
DECFSZ	**f,d**	Decrement f, skip if 0	1 (2)	`0001 011d ffff ffff`	None	6,8
DCFSNZ	**f,d**	Decrement f, skip if not 0	1 (2)	`0010 011d ffff ffff`	None	6,8
INCF	**f,d**	Increment f	1	`0001 010d ffff ffff`	OV,C,DC,Z	
INCFSZ	**f,d**	Increment f, skip if 0	1 (2)	`0001 111d ffff ffff`	None	6,8
INFSNZ	**f,d**	Increment f, skip if not 0	1 (2)	`0010 010d ffff ffff`	None	6,8
IORWF	**f,d**	Inclusive OR WREG with f	1	`0000 100d ffff ffff`	Z	
MOVFP	**f,p**	Move f to p	1	`011p pppp ffff ffff`	None	
MOVPF	**p,f**	Move p to f	1	`010p pppp ffff ffff`	Z	
MOVWF	**f**	Move WREG to f	1	`0000 0001 ffff ffff`	None	
MULWF	**f**	Multiply WREG with f	1	`0011 0100 ffff ffff`	None	9
NEGW	**f,s**	Negate WREG	1	`0010 110s ffff ffff`	OV,C,DC,Z	1,3
NOP	**—**	No Operation	1	`0000 0000 0000 0000`	None	
RLCF	**f,d**	Rotate left f through Carry	1	`0001 101d ffff ffff`	C	
RLNCF	**f,d**	Rotate left f (no carry)	1	`0010 001d ffff ffff`	None	
RRCF	**f,d**	Rotate right f through Carry	1	`0001 100d ffff ffff`	C	
RRNCF	**f,d**	Rotate right f (no carry)	1	`0010 000d ffff ffff`	None	
SETF	**f,s**	Set f	1	`0010 101s ffff ffff`	None	3
SUBWF	**f,d**	Subtract WREG from f	1	`0000 010d ffff ffff`	OV,C,DC,Z	1
SUBWFB	**f,d**	Subtract WREG from f with Borrow	1	`0000 001d ffff ffff`	OV,C,DC,Z	1
SWAPF	**f,d**	Swap f	1	`0001 110d ffff ffff`	None	
TABLRD	**t,i,f**	Table Read	2 (3)	`1010 10ti ffff ffff`	None	7

Legend: Refer to Table 15-1 for opcode field descriptions.

Note 1: 2's Complement method.

2: Unsigned arithmetic.

3: If s = '1', only the file is affected: If s = '0', both the WREG register and the file are affected; If only the Working register (WREG) is required to be affected, then f = WREG must be specified.

4: During an `LCALL`, the contents of PCLATH are loaded into the MSB of the PC and `kkkk kkkk` is loaded into the LSB of the PC (PCL)

5: Multiple cycle instruction for EPROM programming when table pointer selects internal EPROM. The instruction is terminated by an interrupt event. When writing to external program memory, it is a two-cycle instruction.

6: Two-cycle instruction when condition is true, else single cycle instruction.

7: Two-cycle instruction except for `TABLRD` to PCL (program counter low byte) in which case it takes 3 cycles.

8: A "skip" means that instruction fetched during execution of current instruction is not executed, instead an NOP is executed.

9: These instructions are not available on the PIC17C42.

IORWF **Inclusive OR WREG with f**

Syntax: [*label*] IORWF f,d

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

Operation: (WREG) .OR. (f) $\rightarrow$ (dest)

Status Affected: Z

Encoding:

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: [*label*] LCALL k

Operands: $0 \le k \le 255$

Operation: PC + 1 $\rightarrow$ TOS;
k $\rightarrow$ PCL, (PCLATH) $\rightarrow$ PCH

Status Affected: None

Encoding:

1011	0111	kkkk	kkkk

Description: `LCALL` allows an unconditional subroutine call to anywhere within the 64k program memory space.

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.

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Execute	Write register PCL
Forced NOP	NOP	Execute	NOP

Example:

```
MOVLW  HIGH(SUBROUTINE)
MOVPF  WREG, PCLATH
LCALL  LOW(SUBROUTINE)
```

Before Instruction
SUBROUTINE = 16-bit Address
PC = ?

After Instruction
PC = Address (SUBROUTINE)

TLWT — Table Latch Write

Syntax:	[*label*] TLWT t,f
Operands:	0 ≤ f ≤ 255 t ∈ [0,1]
Operation:	If t = 0, f → TBLATL; If t = 1, f → TBLATH
Status Affected:	None
Encoding:	`1010` `01tx` `ffff` `ffff`
Description:	Data from file register 'f' is written into the 16-bit table latch (TBLAT). If t = 1; high byte is written If t = 0; low byte is written This instruction is used in conjunction with `TABLWT` to transfer data from data memory to program memory.
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: `TLWT t, RAM`

Before Instruction
- t = 0
- RAM = 0xB7
- TBLAT = 0x0000 (TBLATH = 0x00) (TBLATL = 0x00)

After Instruction
- RAM = 0xB7
- TBLAT = 0x00B7 (TBLATH = 0x00) (TBLATL = 0xB7)

Before Instruction
- t = 1
- RAM = 0xB7
- TBLAT = 0x0000 (TBLATH = 0x00) (TBLATL = 0x00)

After Instruction
- RAM = 0xB7
- TBLAT = 0xB700 (TBLATH = 0xB7) (TBLATL = 0x00)

TSTFSZ — Test f, skip if 0

Syntax:	[*label*] TSTFSZ f
Operands:	0 ≤ f ≤ 255
Operation:	skip if f = 0
Status Affected:	None
Encoding:	`0011` `0011` `ffff` `ffff`
Description:	If 'f' = 0, the next instruction, fetched during the current instruction execution, is discarded and an NOP is executed 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    TSTFSZ  CNT
NZERO    :
ZERO    :
```

Before Instruction
- PC = Address(HERE)

After Instruction
- If CNT = 0x00,
  - PC = Address (ZERO)
- If CNT ≠ 0x00,
  - PC = Address (NZERO)

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

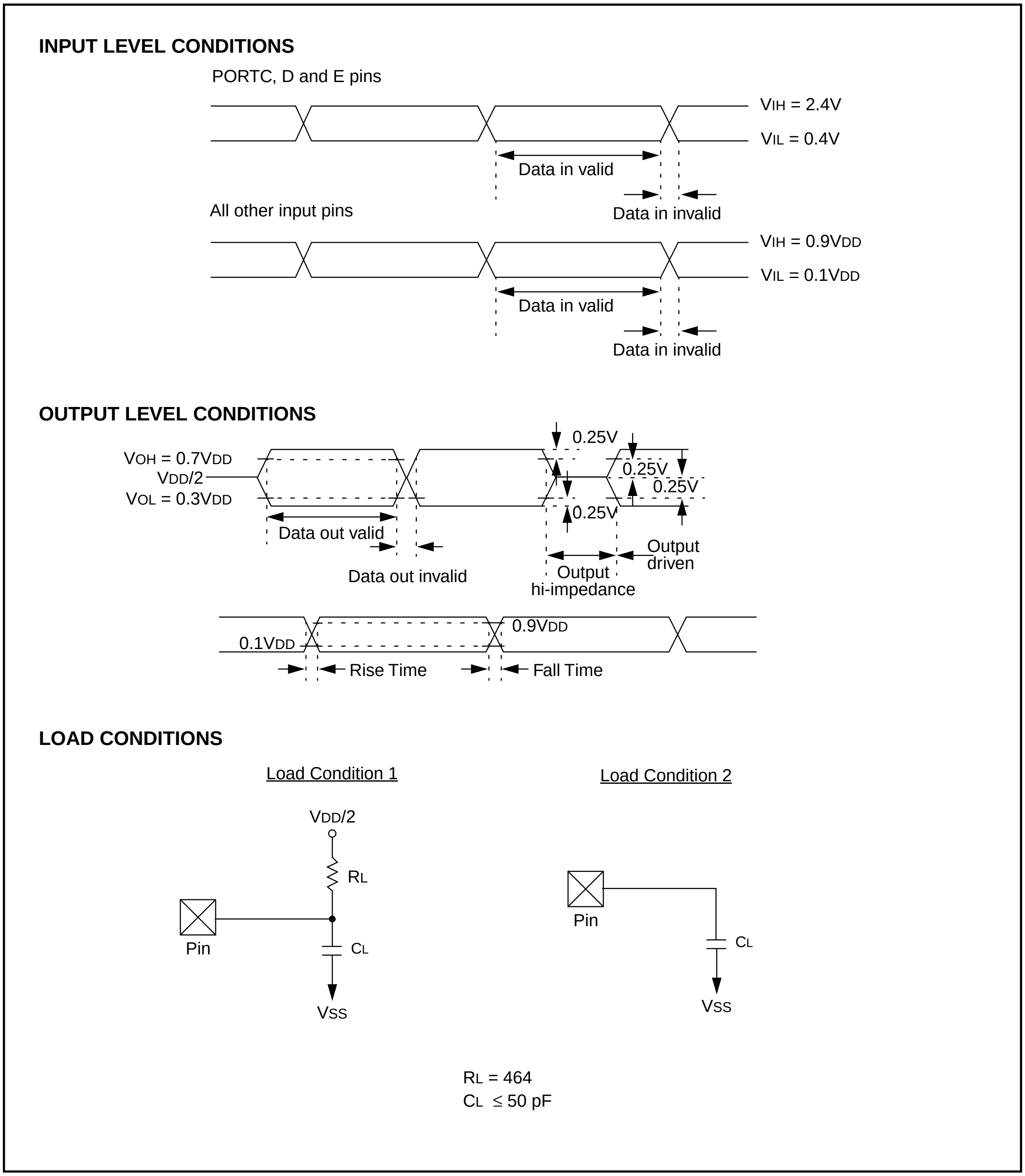
The PICSTART programmer is an easy-to-use, low-cost prototype programmer. It connects to the PC via one of the COM (RS-232) ports. MPLAB Integrated Development Environment software makes using the programmer simple and efficient. PICSTART Plus is not recommended for production programming.

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.

Applicable Devices	42	R42	42A	43	R43	44

FIGURE 17-1: PARAMETER MEASUREMENT INFORMATION

All timings are measure between high and low measurement points as indicated in the figures below.



Applicable Devices	42	R42	42A	43	R43	44

FIGURE 17-11: MEMORY INTERFACE WRITE TIMING

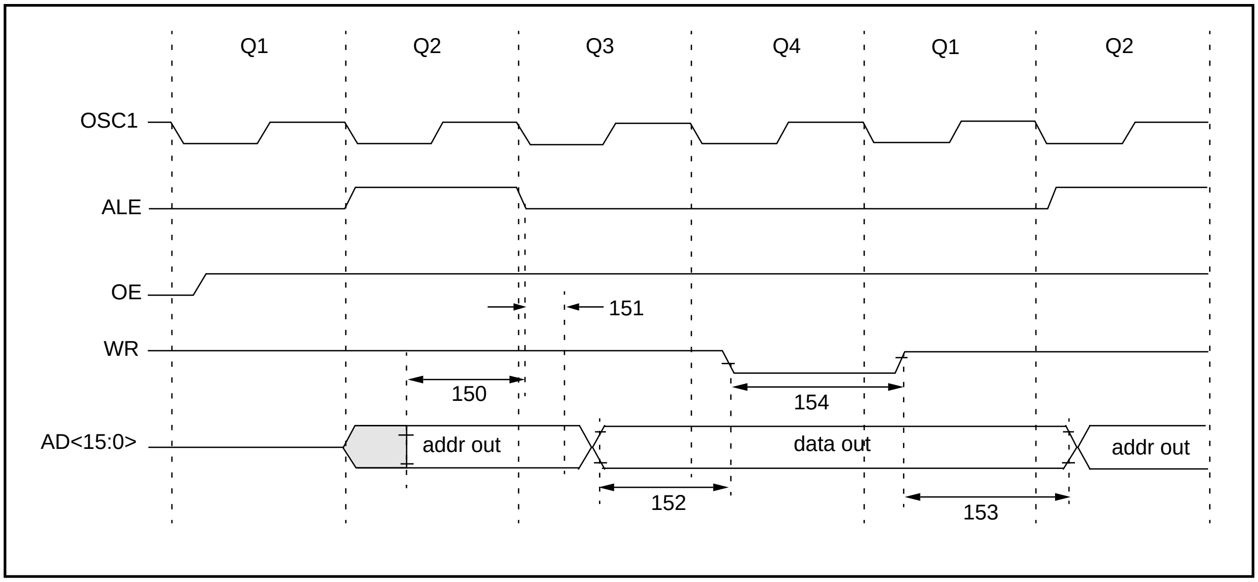


TABLE 17-11: MEMORY INTERFACE WRITE REQUIREMENTS

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	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 is guaranteed by design.

Applicable Devices	42	R42	42A	43	R43	44

FIGURE 18-9: TYPICAL IPD vs. VDD WATCHDOG DISABLED 25°C

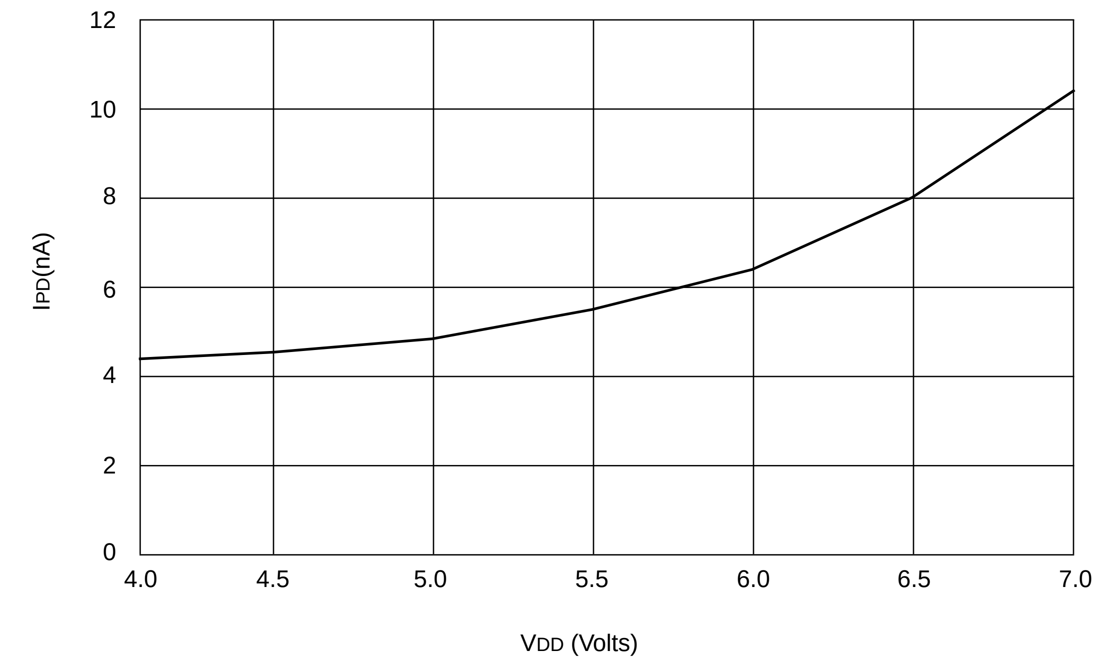
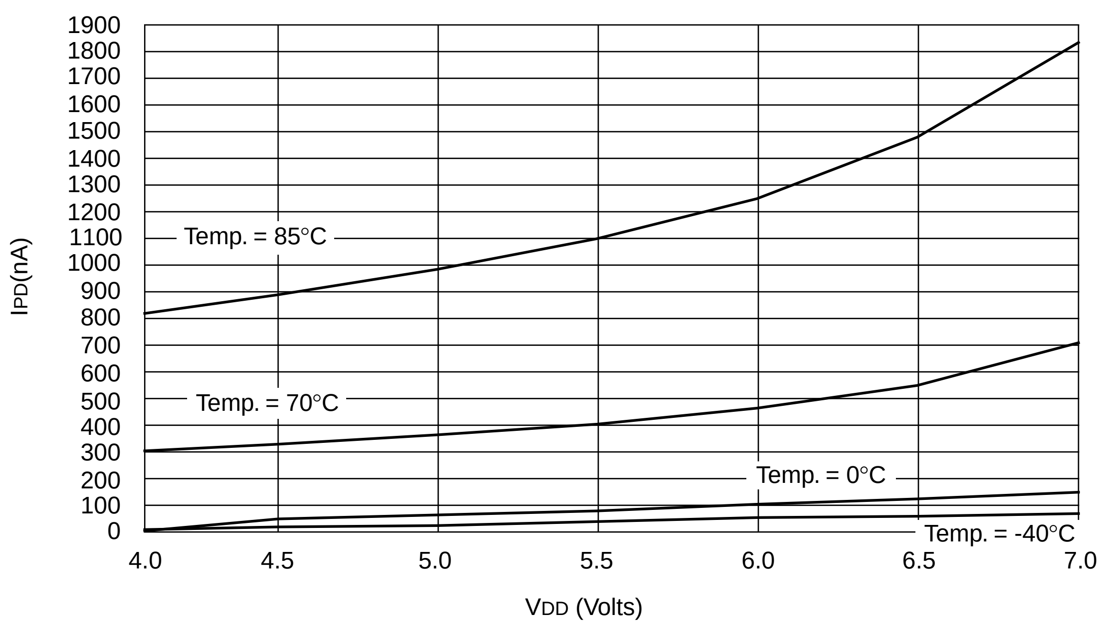


FIGURE 18-10: MAXIMUM IPD vs. VDD WATCHDOG DISABLED



Applicable Devices	42	R42	42A	43	R43	44

FIGURE 18-11: TYPICAL IPD vs. VDD WATCHDOG ENABLED 25°C

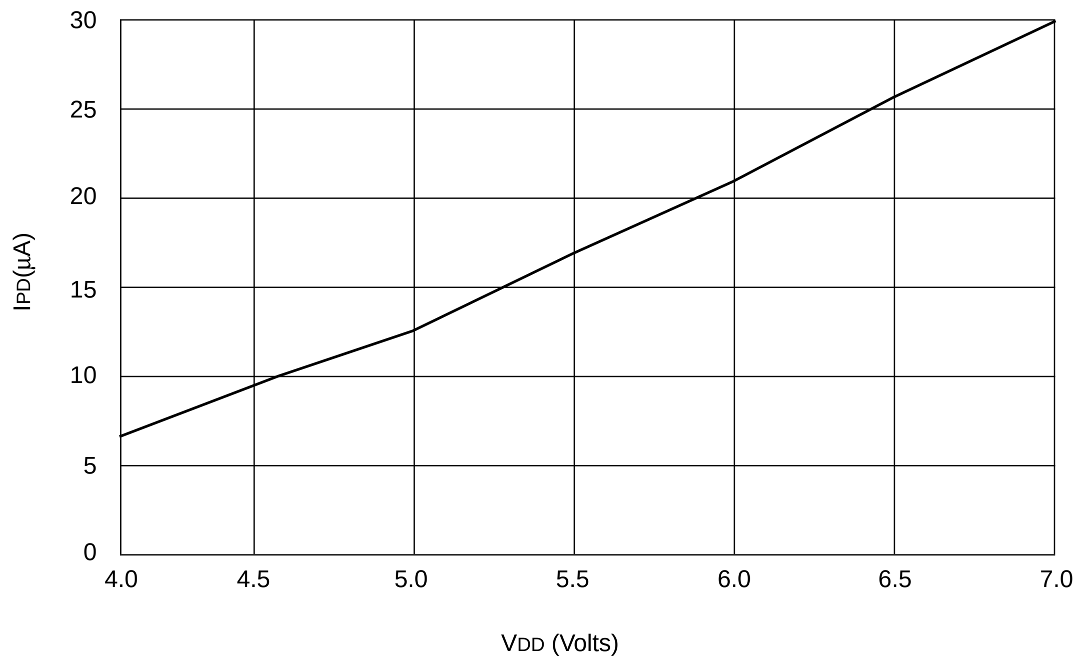
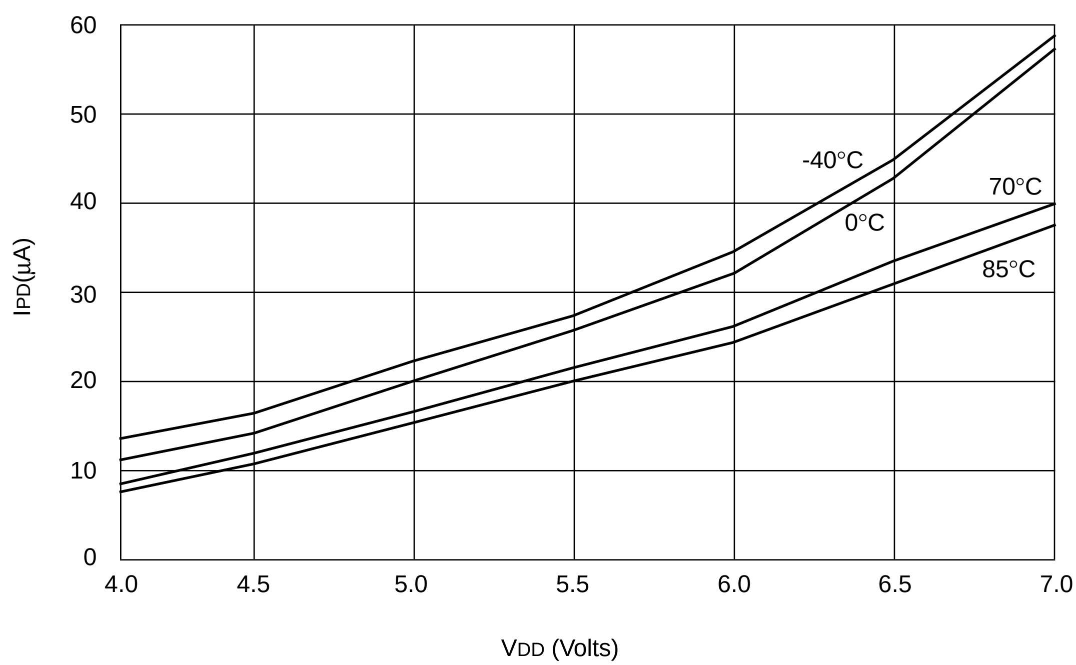


FIGURE 18-12: MAXIMUM IPD vs. VDD WATCHDOG ENABLED



NOTES:

Applicable Devices	42	R42	42A	43	R43	44

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature -40˚C ≤ TA ≤ +85˚C for industrial and 0˚C ≤ TA ≤ +70˚C for commercial Operating voltage VDD range as described in Section 19.1				
Parameter No.	**Sym**	**Characteristic**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
		Output Low Voltage					
D080	V_{OL}	I/O ports (except RA2 and RA3)					I_{OL} = V_{DD}/1.250 mA
			–	–	0.1V_{DD}	V	4.5V ≤ V_{DD} ≤ 6.0V
			–	–	0.1V_{DD} *	V	V_{DD} = 2.5V
D081		with TTL buffer	–	–	0.4	V	I_{OL} = 6 mA, V_{DD} = 4.5V Note 6
D082		RA2 and RA3	–	–	3.0	V	I_{OL} = 60.0 mA, V_{DD} = 6.0V
D083		OSC2/CLKOUT	–	–	0.4	V	I_{OL} = 1 mA, V_{DD} = 4.5V
D084		(RC and EC osc modes)	–	–	0.1V_{DD} *	V	I_{OL} = V_{DD}/5 mA (PIC17LC43/LC44 only)
		Output High Voltage (Note 3)					
D090	V_{OH}	I/O ports (except RA2 and RA3)					I_{OH} = -V_{DD}/2.500 mA
			0.9V_{DD}	–	–	V	4.5V ≤ V_{DD} ≤ 6.0V
			0.9V_{DD} *	–	–	V	V_{DD} = 2.5V
D091		with TTL buffer	2.4	–	–	V	I_{OH} = -6.0 mA, V_{DD}=4.5V Note 6
D092		RA2 and RA3	–	–	12	V	Pulled-up to externally applied voltage
D093		OSC2/CLKOUT	2.4	–	–	V	I_{OH} = -5 mA, V_{DD} = 4.5V
D094		(RC and EC osc modes)	0.9V_{DD} *	–	–	V	I_{OH} = -V_{DD}/5 mA (PIC17LC43/LC44 only)
		Capacitive Loading Specs on Output Pins					
D100	C_{OSC2}	OSC2/CLKOUT pin	–	–	25	pF	In EC or RC osc modes when OSC2 pin is outputting CLKOUT. external clock is used to drive OSC1.
D101	C_{IO}	All I/O pins and OSC2 (in RC mode)	–	–	50	pF	
D102	C_{AD}	System Interface Bus (PORTC, PORTD and PORTE)	–	–	50	pF	In Microprocessor or Extended Microcontroller mode

* 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: In RC oscillator configuration, the OSC1/CLKIN pin is a Schmitt Trigger input. It is not recommended that the PIC17CXX devices be driven with external clock in RC mode.

2: The leakage current on the $\overline{MCLR}$ pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.

3: Negative current is defined as coming out of the pin.

4: These specifications are for the programming of the on-chip program memory EPROM through the use of the table write instructions. The complete programming specifications can be found in: PIC17CXX Programming Specifications (Literature number DS30139).

5: The $\overline{MCLR}$/V_{PP} pin may be kept in this range at times other than programming, but is not recommended.

6: For TTL buffers, the better of the two specifications may be used.

Applicable Devices	42	R42	42A	43	R43	44

FIGURE 20-4: TYPICAL RC OSCILLATOR FREQUENCY vs. VDD

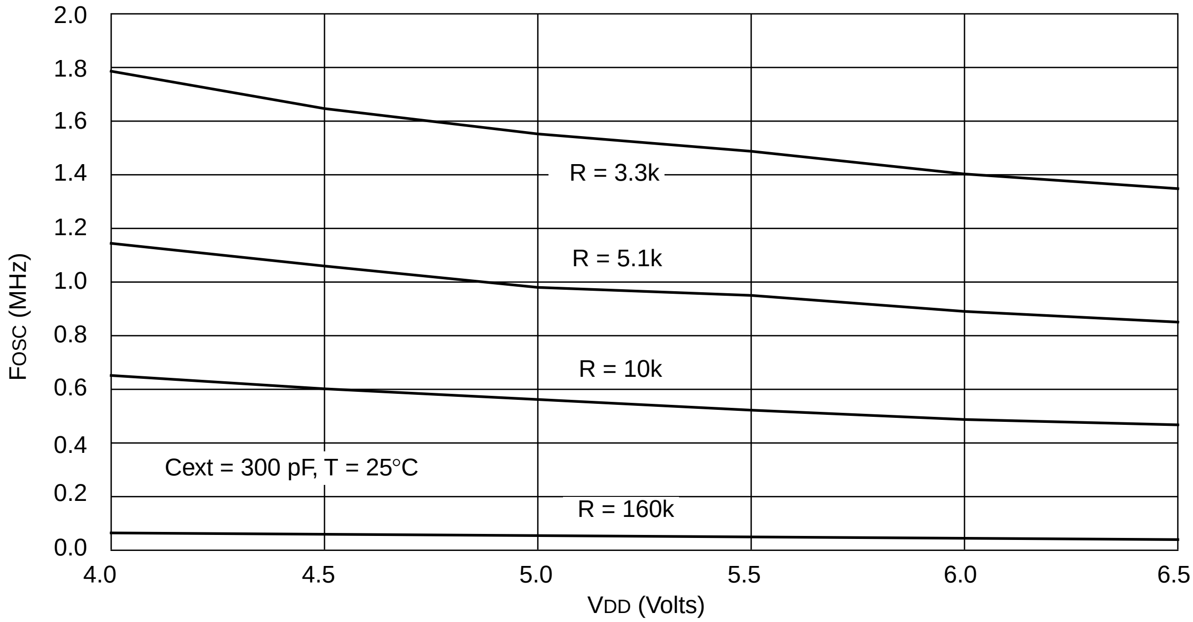


TABLE 20-2: RC OSCILLATOR FREQUENCIES

Cext	Rext	Average Fosc @ 5V, 25°C	
22 pF	10k	3.33 MHz	± 12%
	100k	353 kHz	± 13%
100 pF	3.3k	3.54 MHz	± 10%
	5.1k	2.43 MHz	± 14%
	10k	1.30 MHz	± 17%
	100k	129 kHz	± 10%
300 pF	3.3k	1.54 MHz	± 14%
	5.1k	980 kHz	± 12%
	10k	564 kHz	± 16%
	160k	35 kHz	± 18%

S

T

LIST OF EQUATIONS