

## High-Performance 8-Bit CMOS EPROM/ROM Microcontroller

### Devices included in this data sheet:

- PIC17CR42
- PIC17C42A
- PIC17C43
- PIC17CR43
- PIC17C44
- PIC17C42†

### Microcontroller Core Features:

- Only 58 single word instructions to learn
- All single cycle instructions (121 ns) except for program branches and table reads/writes which are two-cycle
- Operating speed:
  - ★ - DC - 33 MHz clock input
  - DC - 121 ns instruction cycle

Device	Program Memory		Data Memory
	EPROM	ROM	
PIC17CR42	-	2K	232
PIC17C42A	2K	-	232
PIC17C43	4K	-	454
PIC17CR43	-	4K	454
PIC17C44	8K	-	454
PIC17C42†	2K	-	232

- ★ • Hardware Multiplier (Not available on the PIC17C42)
- Interrupt capability
- 16 levels deep hardware stack
- Direct, indirect and relative addressing modes
- Internal/External program memory execution
- 64K x 16 addressable program memory space

### Peripheral Features:

- 33 I/O pins with individual direction control
- High current sink/source for direct LED drive
  - RA2 and RA3 are open drain, high voltage (12V), high current (60 mA), I/O
- Two capture inputs and two PWM outputs
  - Captures are 16-bit, max resolution 160 ns
  - PWM resolution is 1- to 10-bit
- TMR0: 16-bit timer/counter with 8-bit programmable prescaler
- TMR1: 8-bit timer/counter

†NOT recommended for new designs, use 17C42A.

### Pin Diagram



- TMR2: 8-bit timer/counter
- TMR3: 16-bit timer/counter
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI)

### Special Microcontroller Features:

- Power-on Reset (POR), Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Code-protection
- Power saving SLEEP mode
- Selectable oscillator options

### CMOS Technology:

- Low-power, high-speed CMOS EPROM/ROM technology
- Fully static design
- Wide operating voltage range (2.5V to 6.0V)
- Commercial and Industrial Temperature Range
- Low-power consumption
  - < 5 mA @ 5V, 4 MHz
  - 100  $\mu$ A typical @ 4.5V, 32 kHz
  - < 1  $\mu$ A typical standby current @ 5V

# PIC17C4X

## Pin Diagrams Cont'd



## Table of Contents

1.0	Overview .....	5
2.0	PIC17C4X Device Varieties .....	7
3.0	Architectural Overview .....	9
4.0	Reset .....	15
5.0	Interrupts .....	21
6.0	Memory Organization .....	29
7.0	Table Reads and Table Writes.....	43
8.0	Hardware Multiplier .....	49
9.0	I/O Ports .....	53
10.0	Overview of Timer Resources .....	65
11.0	Timer0 .....	67
12.0	Timer1, Timer2, Timer3, PWMs and Captures.....	71
13.0	Universal Synchronous Asynchronous Receiver Transmitter (USART) Module.....	83
14.0	Special Features of the CPU.....	99
15.0	Instruction Set Summary .....	107
16.0	Development Support.....	143
17.0	PIC17C42 Electrical Characteristics .....	147
18.0	PIC17C42 DC and AC Characteristics.....	163
19.0	PIC17CR42/42A/43/R43/44 Electrical Characteristics.....	175
20.0	PIC17CR42/42A/43/R43/44 DC and AC Characteristics .....	193
21.0	Packaging Information.....	205
	Appendix A: Modifications .....	211
	Appendix B: Compatibility.....	211
	Appendix C: What's New .....	212
	Appendix D: What's Changed.....	212
	Appendix E: PIC16/17 Microcontrollers.....	213
	Appendix F: Errata for PIC17C42 Silicon .....	223
	Index .....	226
	PIC17C4X Product Identification System .....	237

For register and module descriptions in this data sheet, device legends show which devices apply to those sections. For example, the legend below shows that some features of only the PIC17C43, PIC17CR43, PIC17C44 are described in this section.

Applicable Devices					
42	R42	42A	43	R43	44

### ***To Our Valued Customers***

We constantly strive to improve the quality of all our products and documentation. We have spent an exceptional amount of time to ensure that these documents are correct. However, we realize that we may have missed a few things. If you find any information that is missing or appears in error from the previous version of the PIC17C4X Data Sheet (Literature Number DS30412B), please use the reader response form in the back of this data sheet to inform us. We appreciate your assistance in making this a better document.

To assist you in the use of this document, Appendix C contains a list of new information in this data sheet, while Appendix D contains information that has changed

# PIC17C4X

---

NOTES:

## 1.0 OVERVIEW

This data sheet covers the PIC17C4X group of the PIC17CXX family of microcontrollers. The following devices are discussed in this data sheet:

- PIC17C42
- PIC17CR42
- PIC17C42A
- PIC17C43
- PIC17CR43
- PIC17C44

The PIC17CR42, PIC17C42A, PIC17C43, PIC17CR43, and PIC17C44 devices include architectural enhancements over the PIC17C42. These enhancements will be discussed throughout this data sheet.

The PIC17C4X devices are 40/44-Pin, EPROM/ROM-based members of the versatile PIC17CXX family of low-cost, high-performance, CMOS, fully-static, 8-bit microcontrollers.

All PIC16/17 microcontrollers employ an advanced RISC architecture. The PIC17CXX has enhanced core features, 16-level deep stack, and multiple internal and external interrupt sources. The separate instruction and data buses of the Harvard architecture allow a 16-bit wide instruction word with a separate 8-bit wide data. The two stage instruction pipeline allows all instructions to execute in a single cycle, except for program branches (which require two cycles). A total of 55 instructions (reduced instruction set) are available in the PIC17C42 and 58 instructions in all the other devices. Additionally, a large register set gives some of the architectural innovations used to achieve a very high performance. For mathematical intensive applications all devices, except the PIC17C42, have a single cycle 8 x 8 Hardware Multiplier.

PIC17CXX microcontrollers typically achieve a 2:1 code compression and a 4:1 speed improvement over other 8-bit microcontrollers in their class.

PIC17C4X devices have up to 454 bytes of RAM and 33 I/O pins. In addition, the PIC17C4X adds several peripheral features useful in many high performance applications including:

- Four timer/counters
- Two capture inputs
- Two PWM outputs
- A Universal Synchronous Asynchronous Receiver Transmitter (USART)

These special features reduce external components, thus reducing cost, enhancing system reliability and reducing power consumption. There are four oscillator options, of which the single pin RC oscillator provides a low-cost solution, the LF oscillator is for low frequency crystals and minimizes power consumption, XT is a standard crystal, and the EC is for external clock input. The SLEEP (power-down) mode offers additional

power saving. The user can wake-up the chip from SLEEP through several external and internal interrupts and device resets.

There are four configuration options for the device operational modes:

- Microprocessor
- Microcontroller
- Extended microcontroller
- Protected microcontroller

The microprocessor and extended microcontroller modes allow up to 64K-words of external program memory.

A highly reliable Watchdog Timer with its own on-chip RC oscillator provides protection against software malfunction.

Table 1-1 lists the features of the PIC17C4X devices.

A UV-erasable Cerdip-packaged version is ideal for code development while the cost-effective One-Time Programmable (OTP) version is suitable for production in any volume.

The PIC17C4X fits perfectly in applications ranging from precise motor control and industrial process control to automotive, instrumentation, and telecom applications. Other applications that require extremely fast execution of complex software programs or the flexibility of programming the software code as one of the last steps of the manufacturing process would also be well suited. The EPROM technology makes customization of application programs (with unique security codes, combinations, model numbers, parameter storage, etc.) fast and convenient. Small footprint package options make the PIC17C4X ideal for applications with space limitations that require high performance. High speed execution, powerful peripheral features, flexible I/O, and low power consumption all at low cost make the PIC17C4X ideal for a wide range of embedded control applications.

### 1.1 Family and Upward Compatibility

Those users familiar with the PIC16C5X and PIC16CXX families of microcontrollers will see the architectural enhancements that have been implemented. These enhancements allow the device to be more efficient in software and hardware requirements. Please refer to Appendix A for a detailed list of enhancements and modifications. Code written for PIC16C5X or PIC16CXX can be easily ported to PIC17CXX family of devices (Appendix B).

### 1.2 Development Support

The PIC17CXX family is supported by a full-featured macro assembler, a software simulator, an in-circuit emulator, a universal programmer, a "C" compiler, and fuzzy logic support tools.

# PIC17C4X

**TABLE 1-1: PIC17CXX FAMILY OF DEVICES**

Features		PIC17C42	PIC17CR42	PIC17C42A	PIC17C43	PIC17CR43	PIC17C44
Maximum Frequency of Operation		25 MHz	33 MHz	33 MHz	33 MHz	33 MHz	33 MHz
Operating Voltage Range		4.5 - 5.5V	2.5 - 6.0V	2.5 - 6.0V	2.5 - 6.0V	2.5 - 6.0V	2.5 - 6.0V
Program Memory x16	(EPROM)	2K	-	2K	4K	-	8K
	(ROM)	-	2K	-	-	4K	-
Data Memory (bytes)		232	232	232	454	454	454
Hardware Multiplier (8 x 8)		-	Yes	Yes	Yes	Yes	Yes
Timer0 (16-bit + 8-bit postscaler)		Yes	Yes	Yes	Yes	Yes	Yes
Timer1 (8-bit)		Yes	Yes	Yes	Yes	Yes	Yes
Timer2 (8-bit)		Yes	Yes	Yes	Yes	Yes	Yes
Timer3 (16-bit)		Yes	Yes	Yes	Yes	Yes	Yes
Capture inputs (16-bit)		2	2	2	2	2	2
PWM outputs (up to 10-bit)		2	2	2	2	2	2
USART/SCI		Yes	Yes	Yes	Yes	Yes	Yes
Power-on Reset		Yes	Yes	Yes	Yes	Yes	Yes
Watchdog Timer		Yes	Yes	Yes	Yes	Yes	Yes
External Interrupts		Yes	Yes	Yes	Yes	Yes	Yes
Interrupt Sources		11	11	11	11	11	11
Program Memory Code Protect		Yes	Yes	Yes	Yes	Yes	Yes
I/O Pins		33	33	33	33	33	33
I/O High Current Capability	Source	25 mA	25 mA	25 mA	25 mA	25 mA	25 mA
	Sink	25 mA <sup>(1)</sup>	25 mA <sup>(1)</sup>	25 mA <sup>(1)</sup>	25 mA <sup>(1)</sup>	25 mA <sup>(1)</sup>	25 mA <sup>(1)</sup>
Package Types		40-pin DIP 44-pin PLCC 44-pin MQFP	40-pin DIP 44-pin PLCC 44-pin MQFP 44-pin TQFP	40-pin DIP 44-pin PLCC 44-pin MQFP 44-pin TQFP	40-pin DIP 44-pin PLCC 44-pin MQFP 44-pin TQFP	40-pin DIP 44-pin PLCC 44-pin MQFP 44-pin TQFP	40-pin DIP 44-pin PLCC 44-pin MQFP 44-pin TQFP

Note 1: Pins RA2 and RA3 can sink up to 60 mA.

## 2.0 PIC17C4X DEVICE VARIETIES

A variety of frequency ranges and packaging options are available. Depending on application and production requirements, the proper device option can be selected using the information in the PIC17C4X Product Selection System section at the end of this data sheet. When placing orders, please use the "PIC17C4X Product Identification System" at the back of this data sheet to specify the correct part number.

For the PIC17C4X family of devices, there are four device "types" as indicated in the device number:

1. **C**, as in PIC17**C**42. These devices have EPROM type memory and operate over the standard voltage range.
2. **LC**, as in PIC17**LC**42. These devices have EPROM type memory, operate over an extended voltage range, and reduced frequency range.
3. **CR**, as in PIC17**CR**42. These devices have ROM type memory and operate over the standard voltage range.
4. **LCR**, as in PIC17**LCR**42. These devices have ROM type memory, operate over an extended voltage range, and reduced frequency range.

### 2.1 UV Erasable Devices

The UV erasable version, offered in CERDIP package, is optimal for prototype development and pilot programs.

The UV erasable version can be erased and reprogrammed to any of the configuration modes. Microchip's PRO MATE™ programmer supports programming of the PIC17C4X. Third party programmers also are available; refer to the *Third Party Guide* for a list of sources.

### 2.2 One-Time-Programmable (OTP) Devices

The availability of OTP devices is especially useful for customers expecting frequent code changes and updates.

The OTP devices, packaged in plastic packages, permit the user to program them once. In addition to the program memory, the configuration bits must also be programmed.

### 2.3 Quick-Turnaround-Production (QTP) Devices

Microchip offers a QTP Programming Service for factory production orders. This service is made available for users who choose not to program a medium to high quantity of units and whose code patterns have stabilized. The devices are identical to the OTP devices but with all EPROM locations and configuration options already programmed by the factory. Certain code and prototype verification procedures apply before production shipments are available. Please contact your local Microchip Technology sales office for more details.

### 2.4 Serialized Quick-Turnaround Production (SQTP<sup>SM</sup>) Devices

Microchip offers a unique programming service where a few user-defined locations in each device are programmed with different serial numbers. The serial numbers may be random, pseudo-random or sequential.

Serial programming allows each device to have a unique number which can serve as an entry-code, password or ID number.

ROM devices do not allow serialization information in the program memory space.

For information on submitting ROM code, please contact your regional sales office.

### 2.5 Read Only Memory (ROM) Devices

Microchip offers masked ROM versions of several of the highest volume parts, thus giving customers a low cost option for high volume, mature products.

For information on submitting ROM code, please contact your regional sales office.

# PIC17C4X

---

NOTES:



## 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 borrow and 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 Math	Unsigned Value 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.

# PIC17C4X

FIGURE 3-1: PIC17C42 BLOCK DIAGRAM



**FIGURE 3-2: PIC17CR42/42A/43/R43/44 BLOCK DIAGRAM**



# PIC17C4X

**TABLE 3-1: PINOUT DESCRIPTIONS**

Name	DIP No.	PLCC No.	QFP No.	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	19	21	37	I	ST	Oscillator input in crystal/resonator or RC oscillator mode. External clock input in external clock mode.
OSC2/CLKOUT	20	22	38	O	—	Oscillator output. Connects to crystal or resonator in crystal oscillator mode. In RC oscillator or external clock modes OSC2 pin outputs CLKOUT which has one fourth the frequency of OSC1 and denotes the instruction cycle rate.
MCLR/VPP	32	35	7	I/P	ST	Master clear (reset) input/Programming Voltage (VPP) input. This is the active low reset input to the chip.
RA0/INT	26	28	44	I	ST	<p>PORTA is a bi-directional I/O Port except for RA0 and RA1 which are input only.</p> <p>RA0/INT can also be selected as an external interrupt input. Interrupt can be configured to be on positive or negative edge.</p> <p>RA1/T0CKI can also be selected as an external interrupt input, and the interrupt can be configured to be on positive or negative edge. RA1/T0CKI can also be selected to be the clock input to the Timer0 timer/counter.</p> <p>High voltage, high current, open drain input/output port pins.</p> <p>High voltage, high current, open drain input/output port pins.</p> <p>RA4/RX/DT can also be selected as the USART (SCI) Asynchronous Receive or USART (SCI) Synchronous Data.</p> <p>RA5/TX/CK can also be selected as the USART (SCI) Asynchronous Transmit or USART (SCI) Synchronous Clock.</p>
RA1/T0CKI	25	27	43	I	ST	
RA2	24	26	42	I/O	ST	
RA3	23	25	41	I/O	ST	
RA4/RX/DT	22	24	40	I/O	ST	
RA5/TX/CK	21	23	39	I/O	ST	
RB0/CAP1	11	13	29	I/O	ST	<p>PORTB is a bi-directional I/O Port with software configurable weak pull-ups.</p> <p>RB0/CAP1 can also be the CAP1 input pin.</p> <p>RB1/CAP2 can also be the CAP2 input pin.</p> <p>RB2/PWM1 can also be the PWM1 output pin.</p> <p>RB3/PWM2 can also be the PWM2 output pin.</p> <p>RB4/TCLK12 can also be the external clock input to Timer1 and Timer2.</p> <p>RB5/TCLK3 can also be the external clock input to Timer3.</p>
RB1/CAP2	12	14	30	I/O	ST	
RB2/PWM1	13	15	31	I/O	ST	
RB3/PWM2	14	16	32	I/O	ST	
RB4/TCLK12	15	17	33	I/O	ST	
RB5/TCLK3	16	18	34	I/O	ST	
RB6	17	19	35	I/O	ST	
RB7	18	20	36	I/O	ST	
RC0/AD0	2	3	19	I/O	TTL	<p>PORTC is a bi-directional I/O Port.</p> <p>This is also the lower half of the 16-bit wide system bus in microprocessor mode or extended microcontroller mode. In multiplexed system bus configuration, these pins are address output as well as data input or output.</p>
RC1/AD1	3	4	20	I/O	TTL	
RC2/AD2	4	5	21	I/O	TTL	
RC3/AD3	5	6	22	I/O	TTL	
RC4/AD4	6	7	23	I/O	TTL	
RC5/AD5	7	8	24	I/O	TTL	
RC6/AD6	8	9	25	I/O	TTL	
RC7/AD7	9	10	26	I/O	TTL	

Legend: I = Input only; O = Output only; I/O = Input/Output; P = Power; — = Not Used; TTL = TTL input; ST = Schmitt Trigger input.

**TABLE 3-1: PINOUT DESCRIPTIONS**

Name	DIP No.	PLCC No.	QFP No.	I/O/P Type	Buffer Type	Description
RD0/AD8	40	43	15	I/O	TTL	PORTD is a bi-directional I/O Port. This is also the upper byte of the 16-bit system bus in microprocessor mode or extended microprocessor mode or extended microcontroller mode. In multiplexed system bus configuration these pins are address output as well as data input or output.
RD1/AD9	39	42	14	I/O	TTL	
RD2/AD10	38	41	13	I/O	TTL	
RD3/AD11	37	40	12	I/O	TTL	
RD4/AD12	36	39	11	I/O	TTL	
RD5/AD13	35	38	10	I/O	TTL	
RD6/AD14	34	37	9	I/O	TTL	
RD7/AD15	33	36	8	I/O	TTL	
RE0/ALE	30	32	4	I/O	TTL	PORTE is a bi-directional I/O Port. In microprocessor mode or extended microcontroller mode, it is the Address Latch Enable (ALE) output. Address should be latched on the falling edge of ALE output. In microprocessor or extended microcontroller mode, it is the Output Enable ( $\overline{OE}$ ) control output (active low). In microprocessor or extended microcontroller mode, it is the Write Enable ( $\overline{WR}$ ) control output (active low).
RE1/ $\overline{OE}$	29	31	3	I/O	TTL	
RE2/ $\overline{WR}$	28	30	2	I/O	TTL	
TEST	27	29	1	I	ST	Test mode selection control input. Always tie to Vss for normal operation.
Vss	10, 31	11, 12, 33, 34	5, 6, 27, 28	P		Ground reference for logic and I/O pins.
VDD	1	1, 44	16, 17	P		Positive supply for logic and I/O pins.

Legend: I = Input only; O = Output only; I/O = Input/Output; P = Power; — = Not Used; TTL = TTL input; ST = Schmitt Trigger input.

# PIC17C4X

## 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.

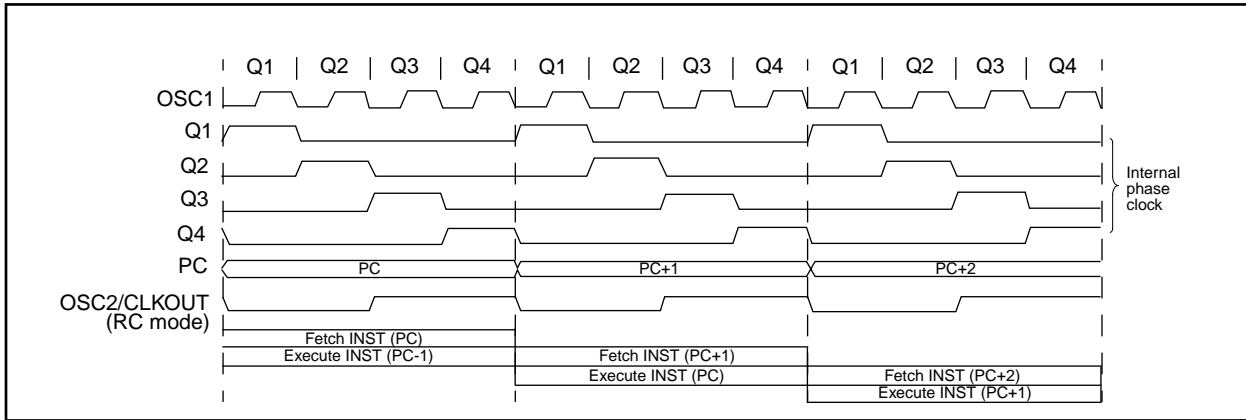
## 3.2 Instruction Flow/Pipelining

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**



## 4.0 RESET

The PIC17CXX differentiates between various kinds of reset:

- Power-on Reset (POR)
- $\overline{\text{MCLR}}$  reset during normal operation
- WDT Reset (normal operation)

Some registers are not affected in any reset condition; their status is unknown on POR and unchanged in any other reset. Most other registers are forced to a "reset state" on Power-on Reset (POR), on  $\overline{\text{MCLR}}$  or WDT Reset and on  $\overline{\text{MCLR}}$  reset during SLEEP. They are not affected by a WDT Reset during SLEEP, since this reset is viewed as the resumption of normal operation. The  $\overline{\text{TO}}$  and  $\overline{\text{PD}}$  bits are set or cleared differently in different reset situations as indicated in Table 4-3. These bits are used in software to determine the nature of reset. See Table 4-4 for a full description of reset states of all registers.

**Note:** While the device is in a reset state, the internal phase clock is held in the Q1 state. Any processor mode that allows external execution will force the RE0/ALE pin as a low output and the RE1/ $\overline{\text{OE}}$  and RE2/ $\overline{\text{WR}}$  pins as high outputs.

A simplified block diagram of the on-chip reset circuit is shown in Figure 4-1.

## 4.1 Power-on Reset (POR), Power-up Timer (PWRT), and Oscillator Start-up Timer (OST)

### 4.1.1 POWER-ON RESET (POR)

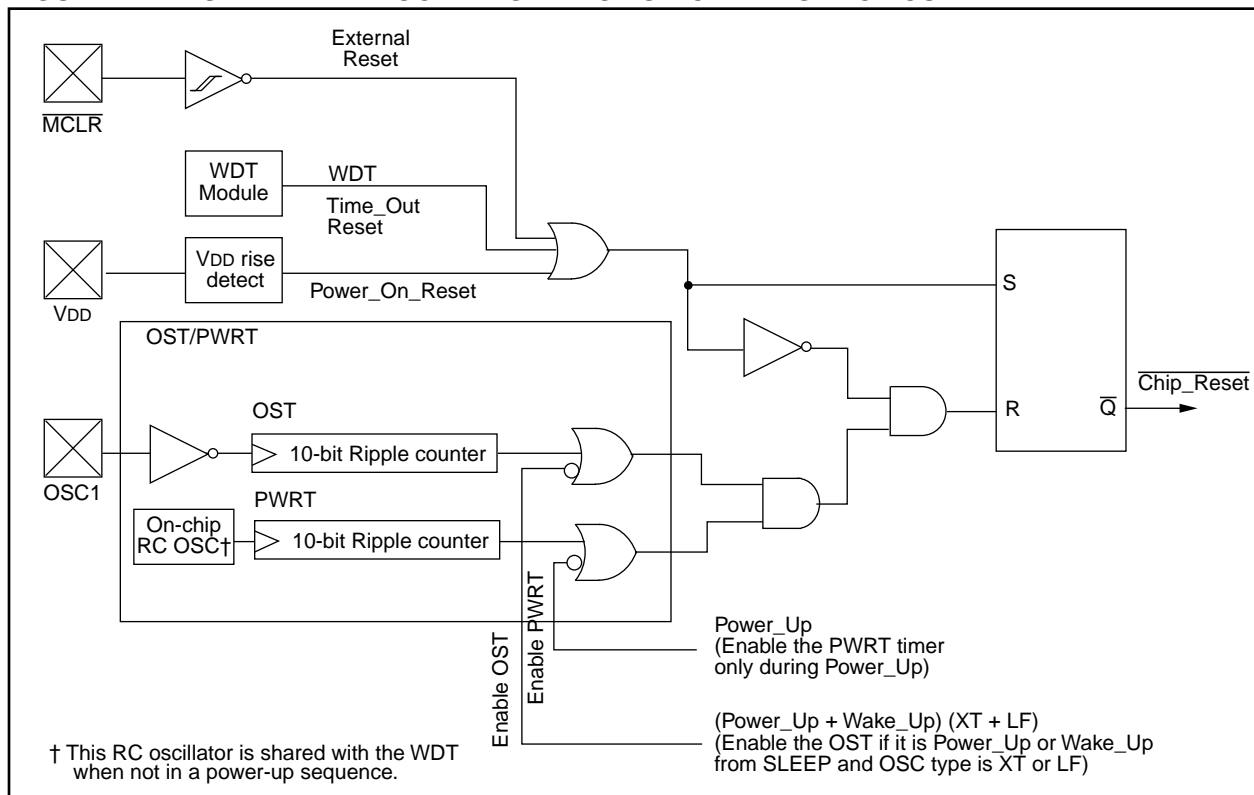
The Power-on Reset circuit holds the device in reset until VDD is above the trip point (in the range of 1.4V - 2.3V). The PIC17C42 does not produce an internal reset when VDD declines. All other devices will produce an internal reset for both rising and falling VDD. To take advantage of the POR, just tie the  $\overline{\text{MCLR/VPP}}$  pin directly (or through a resistor) to VDD. This will eliminate external RC components usually needed to create Power-on Reset. A minimum rise time for VDD is required. See Electrical Specifications for details.

### 4.1.2 POWER-UP TIMER (PWRT)

The Power-up Timer provides a fixed 96 ms time-out (nominal) on power-up. This occurs from rising edge of the POR signal and after the first rising edge of  $\overline{\text{MCLR}}$  (detected high). The Power-up Timer operates on an internal RC oscillator. The chip is kept in RESET as long as the PWRT is active. In most cases the PWRT delay allows the VDD to rise to an acceptable level.

The power-up time delay will vary from chip to chip and to VDD and temperature. See DC parameters for details.

**FIGURE 4-1: SIMPLIFIED BLOCK DIAGRAM OF ON-CHIP RESET CIRCUIT**



# PIC17C4X

## 4.1.3 OSCILLATOR START-UP TIMER (OST)

The Oscillator Start-up Timer (OST) provides a 1024 oscillator cycle (1024Tosc) delay after  $\overline{\text{MCLR}}$  is detected high or a wake-up from SLEEP event occurs.

The OST time-out is invoked only for XT and LF oscillator modes on a Power-on Reset or a Wake-up from SLEEP.

The OST counts the oscillator pulses on the OSC1/CLKIN pin. The counter only starts incrementing after the amplitude of the signal reaches the oscillator input thresholds. This delay allows the crystal oscillator or resonator to stabilize before the device exits reset. The length of time-out is a function of the crystal/resonator frequency.

## 4.1.4 TIME-OUT SEQUENCE

On power-up the time-out sequence is as follows: First the internal POR signal goes high when the POR trip point is reached. If  $\overline{\text{MCLR}}$  is high, then both the OST and PWRT timers start. In general the PWRT time-out is longer, except with low frequency crystals/resonators. The total time-out also varies based on oscillator configuration. Table 4-1 shows the times that are associated with the oscillator configuration. Figure 4-2 and Figure 4-3 display these time-out sequences.

If the device voltage is not within electrical specification at the end of a time-out, the  $\overline{\text{MCLR}}/\text{VPP}$  pin must be held low until the voltage is within the device specification. The use of an external RC delay is sufficient for many of these applications.

**TABLE 4-1: TIME-OUT IN VARIOUS SITUATIONS**

Oscillator Configuration	Power-up	Wake up from SLEEP	MCLR Reset
XT, LF	Greater of: 96 ms or 1024Tosc	1024Tosc	—
EC, RC	Greater of: 96 ms or 1024Tosc	—	—

The time-out sequence begins from the first rising edge of  $\overline{\text{MCLR}}$ .

Table 4-3 shows the reset conditions for some special registers, while Table 4-4 shows the initialization conditions for all the registers. The shaded registers (in Table 4-4) are for all devices except the PIC17C42. In the PIC17C42, the PRODH and PRODL registers are general purpose RAM.

**TABLE 4-2: STATUS BITS AND THEIR SIGNIFICANCE**

$\overline{\text{TO}}$	$\overline{\text{PD}}$	Event
1	1	Power-on Reset, $\overline{\text{MCLR}}$ Reset during normal operation, or CLRWDT instruction executed
1	0	$\overline{\text{MCLR}}$ Reset during SLEEP or interrupt wake-up from SLEEP
0	1	WDT Reset during normal operation
0	0	WDT Reset during SLEEP

In Figure 4-2, Figure 4-3 and Figure 4-4,  $\text{TPWRT} > \text{TOST}$ , as would be the case in higher frequency crystals. For lower frequency crystals, (i.e., 32 kHz) TOST would be greater.

**TABLE 4-3: RESET CONDITION FOR THE PROGRAM COUNTER AND THE CPUSTA REGISTER**

Event		PCH:PCL	CPUSTA	OST Active
Power-on Reset		0000h	--11 11--	Yes
$\overline{\text{MCLR}}$ Reset during normal operation		0000h	--11 11--	No
$\overline{\text{MCLR}}$ Reset during SLEEP		0000h	--11 10--	Yes (2)
WDT Reset during normal operation		0000h	--11 01--	No
WDT Reset during SLEEP (3)		0000h	--11 00--	Yes (2)
Interrupt wake-up from SLEEP	GLINTD is set	PC + 1	--11 10--	Yes (2)
	GLINTD is clear	PC + 1 (1)	--10 10--	Yes (2)

Legend: u = unchanged, x = unknown, - = unimplemented read as '0'.

Note 1: On wake-up, this instruction is executed. The instruction at the appropriate interrupt vector is fetched and then executed.

2: The OST is only active when the Oscillator is configured for XT or LF modes.

3: The Program Counter = 0, that is the device branches to the reset vector. This is different from the mid-range devices.



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



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



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



# PIC17C4X

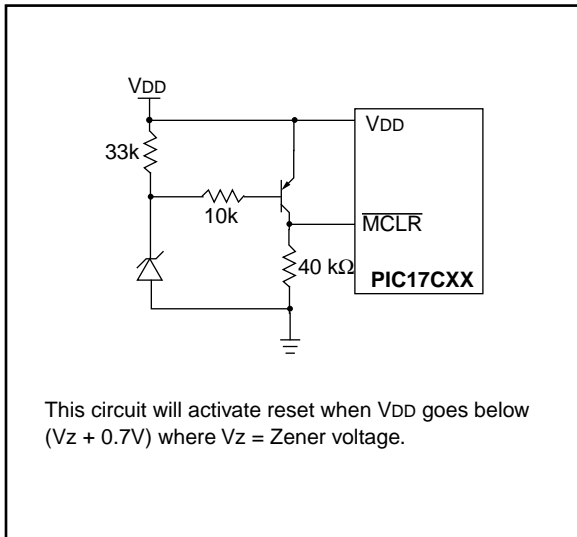
**FIGURE 4-5: OSCILLATOR START-UP TIME**



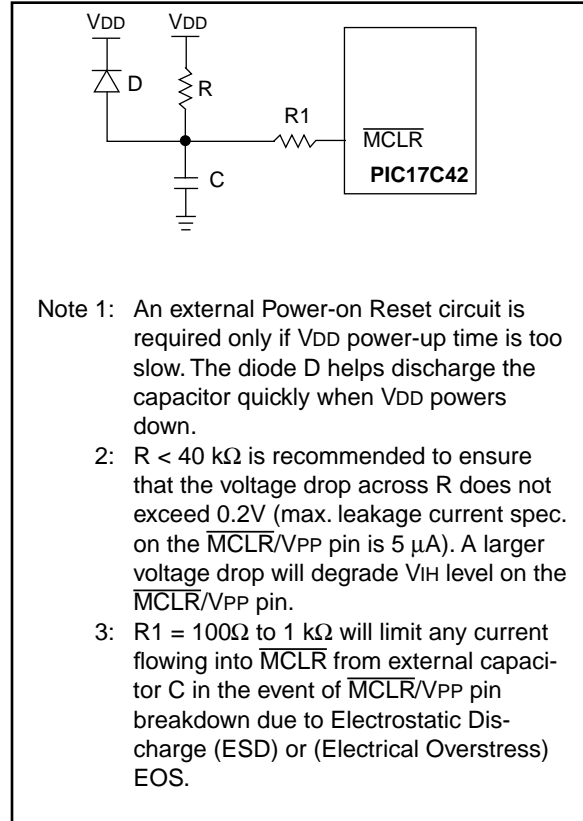
**FIGURE 4-6: USING ON-CHIP POR**



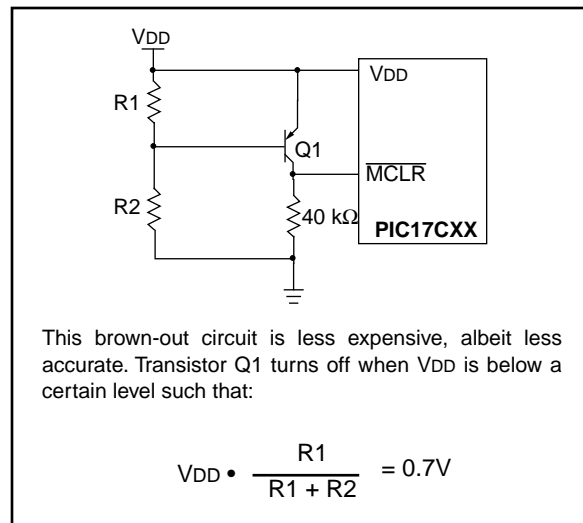
**FIGURE 4-7: BROWN-OUT PROTECTION CIRCUIT 1**



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



**FIGURE 4-9: BROWN-OUT PROTECTION CIRCUIT 2**



**TABLE 4-4: INITIALIZATION CONDITIONS FOR SPECIAL FUNCTION REGISTERS**

Register	Address	Power-on Reset	MCLR Reset WDT Reset	Wake-up from SLEEP through interrupt
<b>Unbanked</b>				
INDF0	00h	0000 0000	0000 0000	0000 0000
FSR0	01h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCL	02h	0000h	0000h	PC + 1 <sup>(2)</sup>
PCLATH	03h	0000 0000	0000 0000	uuuu uuuu
ALUSTA	04h	1111 xxxx	1111 uuuu	1111 uuuu
T0STA	05h	0000 000-	0000 000-	0000 000-
CPUSTA <sup>(3)</sup>	06h	--11 11--	--11 qq--	--uu qq--
INTSTA	07h	0000 0000	0000 0000	uuuu uuuu <sup>(1)</sup>
INDF1	08h	0000 0000	0000 0000	uuuu uuuu
FSR1	09h	xxxx xxxx	uuuu uuuu	uuuu uuuu
WREG	0Ah	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR0L	0Bh	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR0H	0Ch	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBLPTRL <sup>(4)</sup>	0Dh	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBLPTRH <sup>(4)</sup>	0Eh	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBLPTRL <sup>(5)</sup>	0Dh	0000 0000	0000 0000	uuuu uuuu
TBLPTRH <sup>(5)</sup>	0Eh	0000 0000	0000 0000	uuuu uuuu
BSR	0Fh	0000 0000	0000 0000	uuuu uuuu
<b>Bank 0</b>				
PORTA	10h	0-xx xxxx	0-uu uuuu	uuuu uuuu
DDRB	11h	1111 1111	1111 1111	uuuu uuuu
PORTB	12h	xxxx xxxx	uuuu uuuu	uuuu uuuu
RCSTA	13h	0000 -00x	0000 -00u	uuuu -uuu
RCREG	14h	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXSTA	15h	0000 --1x	0000 --1u	uuuu --uu
TXREG	16h	xxxx xxxx	uuuu uuuu	uuuu uuuu
SPBRG	17h	xxxx xxxx	uuuu uuuu	uuuu uuuu
<b>Bank 1</b>				
DDRC	10h	1111 1111	1111 1111	uuuu uuuu
PORTC	11h	xxxx xxxx	uuuu uuuu	uuuu uuuu
DDRD	12h	1111 1111	1111 1111	uuuu uuuu
PORTD	13h	xxxx xxxx	uuuu uuuu	uuuu uuuu
DDRE	14h	---- -111	---- -111	---- -uuu
PORTE	15h	---- -xxx	---- -uuu	---- -uuu
PIR	16h	0000 0010	0000 0010	uuuu uuuu <sup>(1)</sup>
PIE	17h	0000 0000	0000 0000	uuuu uuuu

Legend: u = unchanged, x = unknown, - = unimplemented read as '0', q = value depends on condition.

Note 1: One or more bits in INTSTA, PIR will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GLINTD bit is cleared, the PC is loaded with the interrupt vector.

3: See Table 4-3 for reset value of specific condition.

4: Only applies to the PIC17C42.

5: Does not apply to the PIC17C42.

# PIC17C4X

**TABLE 4-4: INITIALIZATION CONDITIONS FOR SPECIAL FUNCTION REGISTERS (Cont.'d)**

Register	Address	Power-on Reset	MCLR Reset WDT Reset	Wake-up from SLEEP through interrupt
<b>Bank 2</b>				
TMR1	10h	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR2	11h	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR3L	12h	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR3H	13h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PR1	14h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PR2	15h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PR3/CA1L	16h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PR3/CA1H	17h	xxxx xxxx	uuuu uuuu	uuuu uuuu
<b>Bank 3</b>				
PW1DCL	10h	xx-- ----	uu-- ----	uu-- ----
PW2DCL	11h	xx-- ----	uu-- ----	uu-- ----
PW1DCH	12h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PW2DCH	13h	xxxx xxxx	uuuu uuuu	uuuu uuuu
CA2L	14h	xxxx xxxx	uuuu uuuu	uuuu uuuu
CA2H	15h	xxxx xxxx	uuuu uuuu	uuuu uuuu
TCON1	16h	0000 0000	0000 0000	uuuu uuuu
TCON2	17h	0000 0000	0000 0000	uuuu uuuu
<b>Unbanked</b>				
PRODL <sup>(5)</sup>	18h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PRODH <sup>(5)</sup>	19h	xxxx xxxx	uuuu uuuu	uuuu uuuu

Legend: u = unchanged, x = unknown, - = unimplemented read as '0', q = value depends on condition.

Note 1: One or more bits in INTSTA, PIR will be affected (to cause wake-up).

- 2: When the wake-up is due to an interrupt and the GLINTD bit is cleared, the PC is loaded with the interrupt vector.
- 3: See Table 4-3 for reset value of specific condition.
- 4: Only applies to the PIC17C42.
- 5: Does not apply to the PIC17C42.

## 5.0 INTERRUPTS

The PIC17C4X devices have 11 sources of interrupt:

- External interrupt from the RA0/INT pin
- Change on RB7:RB0 pins
- TMR0 Overflow
- TMR1 Overflow
- TMR2 Overflow
- TMR3 Overflow
- USART Transmit buffer empty
- USART Receive buffer full
- Capture1
- Capture2
- T0CKI edge occurred

There are four registers used in the control and status of interrupts. These are:

- CPUSTA
- INTSTA
- PIE
- PIR

The CPUSTA register contains the GLINTD bit. This is the Global Interrupt Disable bit. When this bit is set, all interrupts are disabled. This bit is part of the controller core functionality and is described in the Memory Organization section.

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 the interrupt vector address. There are four interrupt vectors. Each vector address is for a specific interrupt source (except the peripheral interrupts which have the same vector address). These sources are:

- External interrupt from the RA0/INT pin
- TMR0 Overflow
- T0CKI edge occurred
- Any peripheral interrupt

When program execution vectors to one of these interrupt vector addresses (except for the peripheral interrupt address), the interrupt flag bit is automatically cleared. Vectoring to the peripheral interrupt vector address does not automatically clear the source of the interrupt. In the peripheral interrupt service routine, the source(s) of the interrupt can be determined by testing the interrupt flag bits. The interrupt flag bit(s) must be cleared in software before re-enabling interrupts to avoid infinite interrupt requests.

All of the individual interrupt flag bits will be set regardless of the status of their corresponding mask bit or the GLINTD bit.

For external interrupt events, there will be an interrupt latency. For two cycle instructions, the latency could be one instruction cycle longer.

The “return from interrupt” instruction, `RETFIE`, can be used to mark the end of the interrupt service routine. When this instruction is executed, the stack is “POPped”, and the GLINTD bit is cleared (to re-enable interrupts).

**FIGURE 5-1: INTERRUPT LOGIC**



# PIC17C4X

## 5.1 Interrupt Status Register (INTSTA)

The Interrupt Status/Control register (INTSTA) records the individual interrupt requests in flag bits, and contains the individual interrupt enable bits (not for the peripherals).

The PEIF bit is a read only, bit wise OR of all the peripheral flag bits in the PIR register (Figure 5-4).

**Note:** TOIF, INTF, T0CKIF, or PEIF will be set by the specified condition, even if the corresponding interrupt enable bit is clear (interrupt disabled) or the GLINTD bit is set (all interrupts disabled).

Care should be taken when clearing any of the INTSTA register enable bits when interrupts are enabled (GLINTD is clear). If any of the INTSTA flag bits (TOIF, INTF, T0CKIF, or PEIF) are set in the same instruction cycle as the corresponding interrupt enable bit is cleared, the device will vector to the reset address (0x00).

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

**FIGURE 5-2: INTSTA REGISTER (ADDRESS: 07h, UNBANKED)**

	R - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0
	PEIF	T0CKIF	TOIF	INTF	PEIE	T0CKIE	TOIE	INTE
bit7								bit0
<div style="border: 1px solid black; display: inline-block; padding: 2px; margin-top: 5px;">                     R = Readable bit                      W = Writable bit                      - n = Value at POR reset                 </div>								
bit 7:	<b>PEIF:</b> Peripheral Interrupt Flag bit This bit is the OR of all peripheral interrupt flag bits AND'ed with their corresponding enable bits. 1 = A peripheral interrupt is pending 0 = No peripheral interrupt is pending							
bit 6:	<b>T0CKIF:</b> External Interrupt on T0CKI Pin Flag bit This bit is cleared by hardware, when the interrupt logic forces program execution to vector (18h). 1 = The software specified edge occurred on the RA1/T0CKI pin 0 = The software specified edge did not occur on the RA1/T0CKI pin							
bit 5:	<b>TOIF:</b> TMR0 Overflow Interrupt Flag bit This bit is cleared by hardware, when the interrupt logic forces program execution to vector (10h). 1 = TMR0 overflowed 0 = TMR0 did not overflow							
bit 4:	<b>INTF:</b> External Interrupt on INT Pin Flag bit This bit is cleared by hardware, when the interrupt logic forces program execution to vector (08h). 1 = The software specified edge occurred on the RA0/INT pin 0 = The software specified edge did not occur on the RA0/INT pin							
bit 3:	<b>PEIE:</b> Peripheral Interrupt Enable bit This bit enables all peripheral interrupts that have their corresponding enable bits set. 1 = Enable peripheral interrupts 0 = Disable peripheral interrupts							
bit 2:	<b>T0CKIE:</b> External Interrupt on T0CKI Pin Enable bit 1 = Enable software specified edge interrupt on the RA1/T0CKI pin 0 = Disable interrupt on the RA1/T0CKI pin							
bit 1:	<b>TOIE:</b> TMR0 Overflow Interrupt Enable bit 1 = Enable TMR0 overflow interrupt 0 = Disable TMR0 overflow interrupt							
bit 0:	<b>INTE:</b> External Interrupt on RA0/INT Pin Enable bit 1 = Enable software specified edge interrupt on the RA0/INT pin 0 = Disable software specified edge interrupt on the RA0/INT pin							

## 5.2 Peripheral Interrupt Enable Register (PIE)

This register contains the individual flag bits for the Peripheral interrupts.

**FIGURE 5-3: PIE REGISTER (ADDRESS: 17h, BANK 1)**

R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0
RBIE	TMR3IE	TMR2IE	TMR1IE	CA2IE	CA1IE	TXIE	RCIE
						bit0	
<p>bit 7: <b>RBIE:</b> PORTB Interrupt on Change Enable bit            1 = Enable PORTB interrupt on change            0 = Disable PORTB interrupt on change</p> <p>bit 6: <b>TMR3IE:</b> Timer3 Interrupt Enable bit            1 = Enable Timer3 interrupt            0 = Disable Timer3 interrupt</p> <p>bit 5: <b>TMR2IE:</b> Timer2 Interrupt Enable bit            1 = Enable Timer2 interrupt            0 = Disable Timer2 interrupt</p> <p>bit 4: <b>TMR1IE:</b> Timer1 Interrupt Enable bit            1 = Enable Timer1 interrupt            0 = Disable Timer1 interrupt</p> <p>bit 3: <b>CA2IE:</b> Capture2 Interrupt Enable bit            1 = Enable Capture interrupt on RB1/CAP2 pin            0 = Disable Capture interrupt on RB1/CAP2 pin</p> <p>bit 2: <b>CA1IE:</b> Capture1 Interrupt Enable bit            1 = Enable Capture interrupt on RB2/CAP1 pin            0 = Disable Capture interrupt on RB2/CAP1 pin</p> <p>bit 1: <b>TXIE:</b> USART Transmit Interrupt Enable bit            1 = Enable Transmit buffer empty interrupt            0 = Disable Transmit buffer empty interrupt</p> <p>bit 0: <b>RCIE:</b> USART Receive Interrupt Enable bit            1 = Enable Receive buffer full interrupt            0 = Disable Receive buffer full interrupt</p>							

R = Readable bit  
 W = Writable bit  
 -n = Value at POR reset

# PIC17C4X

## 5.3 Peripheral Interrupt Request Register (PIR)

This register contains the individual flag bits for the peripheral interrupts.

**Note:** These bits will be set by the specified condition, even if the corresponding interrupt enable bit is cleared (interrupt disabled), or the GLINTD bit is set (all interrupts disabled). Before enabling an interrupt, the user may wish to clear the interrupt flag to ensure that the program does not immediately branch to the peripheral interrupt service routine.

**FIGURE 5-4: PIR REGISTER (ADDRESS: 16h, BANK 1)**

	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R - 1	R - 0
	RBIF	TMR3IF	TMR2IF	TMR1IF	CA2IF	CA1IF	TXIF	RCIF
bit7								bit0
<div style="float: right; border: 1px solid black; padding: 2px; font-size: small;">                     R = Readable bit                      W = Writable bit                      -n = Value at POR reset                 </div>								
bit 7:	<b>RBIF:</b> PORTB Interrupt on Change Flag bit 1 = One of the PORTB inputs changed (Software must end the mismatch condition) 0 = None of the PORTB inputs have changed							
bit 6:	<b>TMR3IF:</b> Timer3 Interrupt Flag bit If Capture1 is enabled ( $CA1/\overline{PR3} = 1$ ) 1 = Timer3 overflowed 0 = Timer3 did not overflow  If Capture1 is disabled ( $CA1/\overline{PR3} = 0$ ) 1 = Timer3 value has rolled over to 0000h from equalling the period register (PR3H:PR3L) value 0 = Timer3 value has not rolled over to 0000h from equalling the period register (PR3H:PR3L) value							
bit 5:	<b>TMR2IF:</b> Timer2 Interrupt Flag bit 1 = Timer2 value has rolled over to 0000h from equalling the period register (PR2) value 0 = Timer2 value has not rolled over to 0000h from equalling the period register (PR2) value							
bit 4:	<b>TMR1IF:</b> Timer1 Interrupt Flag bit If Timer1 is in 8-bit mode (T16 = 0) 1 = Timer1 value has rolled over to 0000h from equalling the period register (PR) value 0 = Timer1 value has not rolled over to 0000h from equalling the period register (PR2) value  If Timer1 is in 16-bit mode (T16 = 1) 1 = TMR1:TMR2 value has rolled over to 0000h from equalling the period register (PR1:PR2) value 0 = TMR1:TMR2 value has not rolled over to 0000h from equalling the period register (PR1:PR2) value							
bit 3:	<b>CA2IF:</b> Capture2 Interrupt Flag bit 1 = Capture event occurred on RB1/CAP2 pin 0 = Capture event did not occur on RB1/CAP2 pin							
bit 2:	<b>CA1IF:</b> Capture1 Interrupt Flag bit 1 = Capture event occurred on RB0/CAP1 pin 0 = Capture event did not occur on RB0/CAP1 pin							
bit 1:	<b>TXIF:</b> USART Transmit Interrupt Flag bit 1 = Transmit buffer is empty 0 = Transmit buffer is full							
bit 0:	<b>RCIF:</b> USART Receive Interrupt Flag bit 1 = Receive buffer is full 0 = Receive buffer is empty							



## 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 (TOIF)	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
    
```

## 5.5 RA0/INT Interrupt

The external interrupt on the RA0/INT pin is edge triggered. Either the rising edge, if INTEDG bit (T0STA<7>) is set, or the falling edge, if INTEDG bit is clear. When a valid edge appears on the RA0/INT pin, the INTF bit (INTSTA<4>) is set. This interrupt can be disabled by clearing the INTE control bit (INTSTA<0>). The INT interrupt can wake the processor from SLEEP. See Section 14.4 for details on SLEEP operation.

## 5.6 TMR0 Interrupt

An overflow (FFFFh → 0000h) in TMR0 will set the T0IF (INTSTA<5>) bit. The interrupt can be enabled/disabled by setting/clearing the T0IE control bit (INTSTA<1>). For operation of the Timer0 module, see Section 11.0.

## 5.7 T0CKI Interrupt

The external interrupt on the RA1/T0CKI pin is edge triggered. Either the rising edge, if the T0SE bit (T0STA<6>) is set, or the falling edge, if the T0SE bit is clear. When a valid edge appears on the RA1/T0CKI pin, the T0CKIF bit (INTSTA<6>) is set. This interrupt can be disabled by clearing the T0CKIE control bit (INTSTA<2>). The T0CKI interrupt can wake up the processor from SLEEP. See Section 14.4 for details on SLEEP operation.

## 5.8 Peripheral Interrupt

The peripheral interrupt flag indicates that at least one of the peripheral interrupts occurred (PEIF is set). The PEIF bit is a read only bit, and is a bit wise OR of all the flag bits in the PIR register AND'ed with the corresponding enable bits in the PIE register. Some of the peripheral interrupts can wake the processor from SLEEP. See Section 14.4 for details on SLEEP operation.

**FIGURE 5-5: INT PIN / T0CKI PIN INTERRUPT TIMING**



## 5.9 Context Saving During Interrupts

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

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

### EXAMPLE 5-1: SAVING STATUS AND WREG IN RAM

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

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

# PIC17C4X

---

NOTES:

## 6.0 MEMORY ORGANIZATION

There are two memory blocks in the PIC17C4X; program memory and data memory. Each block has its own bus, so that access to each block can occur during the same oscillator cycle.

The data memory can further be broken down into General Purpose RAM and the Special Function Registers (SFRs). The operation of the SFRs that control the "core" are described here. The SFRs used to control the peripheral modules are described in the section discussing each individual peripheral module.

### 6.1 Program Memory Organization

PIC17C4X devices have a 16-bit program counter capable of addressing a 64K x 16 program memory space. The reset vector is at 0000h and the interrupt vectors are at 0008h, 0010h, 0018h, and 0020h (Figure 6-1).

#### 6.1.1 PROGRAM MEMORY OPERATION

The PIC17C4X can operate in one of four possible program memory configurations. The configuration is selected by two configuration bits. The possible modes are:

- Microprocessor
- Microcontroller
- Extended Microcontroller
- Protected Microcontroller

The microcontroller and protected microcontroller modes only allow internal execution. Any access beyond the program memory reads unknown data. The protected microcontroller mode also enables the code protection feature.

The extended microcontroller mode accesses both the internal program memory as well as external program memory. Execution automatically switches between internal and external memory. The 16-bits of address allow a program memory range of 64K-words.

The microprocessor mode only accesses the external program memory. The on-chip program memory is ignored. The 16-bits of address allow a program memory range of 64K-words. Microprocessor mode is the default mode of an unprogrammed device.

The different modes allow different access to the configuration bits, test memory, and boot ROM. Table 6-1 lists which modes can access which areas in memory. Test Memory and Boot Memory are not required for normal operation of the device. Care should be taken to ensure that no unintended branches occur to these areas.

**FIGURE 6-1: PROGRAM MEMORY MAP AND STACK**



# PIC17C4X

**TABLE 6-1: MODE MEMORY ACCESS**

Operating Mode	Internal Program Memory	Configuration Bits, Test Memory, Boot ROM
Microprocessor	No Access	No Access
Microcontroller	Access	Access
Extended Microcontroller	Access	No Access
Protected Microcontroller	Access	Access

The PIC17C4X can operate in modes where the program memory is off-chip. They are the microprocessor and extended microcontroller modes. The microprocessor mode is the default for an unprogrammed device.

Regardless of the processor mode, data memory is always on-chip.

**FIGURE 6-2: MEMORY MAP IN DIFFERENT MODES**



## 6.1.2 EXTERNAL MEMORY INTERFACE

When either microprocessor or extended microcontroller mode is selected, PORTC, PORTD and PORTE are configured as the system bus. PORTC and PORTD are the multiplexed address/data bus and PORTE is for the control signals. External components are needed to demultiplex the address and data. This can be done as shown in Figure 6-4. The waveforms of address and data are shown in Figure 6-3. For complete timings, please refer to the electrical specification section.

**FIGURE 6-3: EXTERNAL PROGRAM MEMORY ACCESS WAVEFORMS**



The system bus requires that there is no bus conflict (minimal leakage), so the output value (address) will be capacitively held at the desired value.

As the speed of the processor increases, external EPROM memory with faster access time must be used. Table 6-2 lists external memory speed requirements for a given PIC17C4X device frequency.

In extended microcontroller mode, when the device is executing out of internal memory, the control signals will continue to be active. That is, they indicate the action that is occurring in the internal memory. The external memory access is ignored.

This following selection is for use with Microchip EPROMs. For interfacing to other manufacturers memory, please refer to the electrical specifications of the desired PIC17C4X device, as well as the desired memory device to ensure compatibility.

**TABLE 6-2: EPROM MEMORY ACCESS TIME ORDERING SUFFIX**

PIC17C4X Oscillator Frequency	Instruction Cycle Time (Tcy)	EPROM Suffix	
		PIC17C42	PIC17C43 PIC17C44
8 MHz	500 ns	-25	-25
16 MHz	250 ns	-12	-15
20 MHz	200 ns	-90	-10
25 MHz	160 ns	N.A.	-70
33 MHz	121 ns	N.A.	(1)

Note 1: The access times for this requires the use of fast SRAMS.

**Note:** The external memory interface is not supported for the LC devices.

**FIGURE 6-4: TYPICAL EXTERNAL PROGRAM MEMORY CONNECTION DIAGRAM**



## 6.2 Data Memory Organization

Data memory is partitioned into two areas. The first is the General Purpose Registers (GPR) area, while the second is the Special Function Registers (SFR) area. The SFRs control the operation of the device.

Portions of data memory are banked, this is for both areas. The GPR area is banked to allow greater than 232 bytes of general purpose RAM. SFRs are for the registers that control the peripheral functions. Banking requires the use of control bits for bank selection. These control bits are located in the Bank Select Register (BSR). If an access is made to a location outside this banked region, the BSR bits are ignored. Figure 6-5 shows the data memory map organization for the PIC17C42 and Figure 6-6 for all of the other PIC17C4X devices.

Instructions `MOVPF` and `MOVFP` provide the means to move values from the peripheral area ("P") to any location in the register file ("F"), and vice-versa. The definition of the "P" range is from 0h to 1Fh, while the "F" range is 0h to FFh. The "P" range has six more locations than peripheral registers (eight locations for the PIC17C42 device) which can be used as General Purpose Registers. This can be useful in some applications where variables need to be copied to other locations in the general purpose RAM (such as saving status information during an interrupt).

The entire data memory can be accessed either directly or indirectly through file select registers FSR0 and FSR1 (Section 6.4). Indirect addressing uses the appropriate control bits of the BSR for accesses into the banked areas of data memory. The BSR is explained in greater detail in Section 6.8.

### 6.2.1 GENERAL PURPOSE REGISTER (GPR)

All devices have some amount of GPR area. The GPRs are 8-bits wide. When the GPR area is greater than 232, it must be banked to allow access to the additional memory space.

Only the PIC17C43 and PIC17C44 devices have banked memory in the GPR area. To facilitate switching between these banks, the `MOVLR` bank instruction has been added to the instruction set. GPRs are not initialized by a Power-on Reset and are unchanged on all other resets.

### 6.2.2 SPECIAL FUNCTION REGISTERS (SFR)

The SFRs are used by the CPU and peripheral functions to control the operation of the device (Figure 6-5 and Figure 6-6). These registers are static RAM.

The SFRs can be classified into two sets, those associated with the "core" function and those related to the peripheral functions. Those registers related to the "core" are described here, while those related to a peripheral feature are described in the section for each peripheral feature.

The peripheral registers are in the banked portion of memory, while the core registers are in the unbanked region. To facilitate switching between the peripheral banks, the `MOVLB` bank instruction has been provided.



**FIGURE 6-5: PIC17C42 REGISTER FILE MAP**

Addr	Unbanked			
00h	INDF0			
01h	FSR0			
02h	PCL			
03h	PCLATH			
04h	ALUSTA			
05h	T0STA			
06h	CPUSTA			
07h	INTSTA			
08h	INDF1			
09h	FSR1			
0Ah	WREG			
0Bh	TMR0L			
0Ch	TMR0H			
0Dh	TBLPTRL			
0Eh	TBLPTRH			
0Fh	BSR			
	Bank 0	Bank 1 (1)	Bank 2 (1)	Bank 3 (1)
10h	PORTA	DDRC	TMR1	PW1DCL
11h	DDRB	PORTC	TMR2	PW2DCL
12h	PORTB	DDRD	TMR3L	PW1DCH
13h	RCSTA	PORTD	TMR3H	PW2DCH
14h	RCREG	DDRE	PR1	CA2L
15h	TXSTA	PORTE	PR2	CA2H
16h	TXREG	PIR	PR3L/CA1L	TCON1
17h	SPBRG	PIE	PR3H/CA1H	TCON2
18h				
1Fh	General Purpose RAM			
20h				
FFh				

Note 1: SFR file locations 10h - 17h are banked. All other SFRs ignore the Bank Select Register (BSR) bits.

**FIGURE 6-6: PIC17CR42/42A/43/R43/44 REGISTER FILE MAP**

Addr	Unbanked			
00h	INDF0			
01h	FSR0			
02h	PCL			
03h	PCLATH			
04h	ALUSTA			
05h	T0STA			
06h	CPUSTA			
07h	INTSTA			
08h	INDF1			
09h	FSR1			
0Ah	WREG			
0Bh	TMR0L			
0Ch	TMR0H			
0Dh	TBLPTRL			
0Eh	TBLPTRH			
0Fh	BSR			
	Bank 0	Bank 1 (1)	Bank 2 (1)	Bank 3 (1)
10h	PORTA	DDRC	TMR1	PW1DCL
11h	DDRB	PORTC	TMR2	PW2DCL
12h	PORTB	DDRD	TMR3L	PW1DCH
13h	RCSTA	PORTD	TMR3H	PW2DCH
14h	RCREG	DDRE	PR1	CA2L
15h	TXSTA	PORTE	PR2	CA2H
16h	TXREG	PIR	PR3L/CA1L	TCON1
17h	SPBRG	PIE	PR3H/CA1H	TCON2
18h				
19h				
1Ah				
1Fh	General Purpose RAM (2)			
20h			General Purpose RAM (2)	
FFh				

Note 1: SFR file locations 10h - 17h are banked. All other SFRs ignore the Bank Select Register (BSR) bits.

2: General Purpose Registers (GPR) locations 20h - FFh and 120h - 1FFh are banked. All other GPRs ignore the Bank Select Register (BSR) bits.

# PIC17C4X

**TABLE 6-3: SPECIAL FUNCTION REGISTERS**

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 (3)
<b>Unbanked</b>											
00h	INDF0	Uses contents of FSR0 to address data memory (not a physical register)								----	----
01h	FSR0	Indirect data memory address pointer 0								xxxx xxxx	uuuu uuuu
02h	PCL	Low order 8-bits of PC								0000 0000	0000 0000
03h <sup>(1)</sup>	PCLATH	Holding register for upper 8-bits of PC								0000 0000	uuuu uuuu
04h	ALUSTA	FS3	FS2	FS1	FS0	OV	Z	DC	C	1111 xxxx	1111 uuuu
05h	T0STA	INTEDG	T0SE	T0CS	PS3	PS2	PS1	PS0	—	0000 000-	0000 000-
06h <sup>(2)</sup>	CPUSTA	—	—	STKAV	GLINTD	$\overline{TO}$	$\overline{PD}$	—	—	--11 11--	--11 qq--
07h	INTSTA	PEIF	T0CKIF	T0IF	INTF	PEIE	T0CKIE	T0IE	INTE	0000 0000	0000 0000
08h	INDF1	Uses contents of FSR1 to address data memory (not a physical register)								----	----
09h	FSR1	Indirect data memory address pointer 1								xxxx xxxx	uuuu uuuu
0Ah	WREG	Working register								xxxx xxxx	uuuu uuuu
0Bh	TMR0L	TMR0 register; low byte								xxxx xxxx	uuuu uuuu
0Ch	TMR0H	TMR0 register; high byte								xxxx xxxx	uuuu uuuu
0Dh	TBLPTRL	Low byte of program memory table pointer								(4)	(4)
0Eh	TBLPTRH	High byte of program memory table pointer								(4)	(4)
0Fh	BSR	Bank select register								0000 0000	0000 0000
<b>Bank 0</b>											
10h	PORTA	$\overline{RBP}$	—	RA5	RA4	RA3	RA2	RA1/T0CKI	RA0/INT	0-xx xxxx	0-uu uuuu
11h	DDRB	Data direction register for PORTB								1111 1111	1111 1111
12h	PORTB	PORTB data latch								xxxx xxxx	uuuu uuuu
13h	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00u
14h	RCREG	Serial port receive register								xxxx xxxx	uuuu uuuu
15h	TXSTA	CSRC	TX9	TXEN	SYNC	—	—	TRMT	TX9D	0000 --1x	0000 --1u
16h	TXREG	Serial port transmit register								xxxx xxxx	uuuu uuuu
17h	SPBRG	Baud rate generator register								xxxx xxxx	uuuu uuuu
<b>Bank 1</b>											
10h	DDRC	Data direction register for PORTC								1111 1111	1111 1111
11h	PORTC	RC7/AD7	RC6/AD6	RC5/AD5	RC4/AD4	RC3/AD3	RC2/AD2	RC1/AD1	RC0/AD0	xxxx xxxx	uuuu uuuu
12h	DDRD	Data direction register for PORTD								1111 1111	1111 1111
13h	PORTD	RD7/AD15	RD6/AD14	RD5/AD13	RD4/AD12	RD3/AD11	RD2/AD10	RD1/AD9	RD0/AD8	xxxx xxxx	uuuu uuuu
14h	DDRE	Data direction register for PORTE								---- -111	---- -111
15h	PORTE	—	—	—	—	—	RE2/ $\overline{WR}$	RE1/ $\overline{OE}$	RE0/ALE	---- -xxx	---- -uuu
16h	PIR	RBIF	TMR3IF	TMR2IF	TMR1IF	CA2IF	CA1IF	TXIF	RCIF	0000 0010	0000 0010
17h	PIE	RBIE	TMR3IE	TMR2IE	TMR1IE	CA2IE	CA1IE	TXIE	RCIE	0000 0000	0000 0000

Legend: x = unknown, u = unchanged, - = unimplemented read as '0', q - value depends on condition. Shaded cells are unimplemented, read as '0'.

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for PC<15:8> whose contents are updated from or transferred to the upper byte of the program counter.

2: The  $\overline{TO}$  and  $\overline{PD}$  status bits in CPUSTA are not affected by a MCLR reset.

3: Other (non power-up) resets include: external reset through MCLR and the Watchdog Timer Reset.

4: The following values are for both TBLPTRL and TBLPTRH:

All PIC17C4X devices (Power-on Reset 0000 0000) and (All other resets 0000 0000) except the PIC17C42 (Power-on Reset xxxx xxxx) and (All other resets uuuu uuuu)

5: The PRODL and PRODH registers are not implemented on the PIC17C42.

**TABLE 6-3: SPECIAL FUNCTION REGISTERS (Cont.'d)**

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 (3)
<b>Bank 2</b>											
10h	TMR1	Timer1								xxxx xxxx	uuuu uuuu
11h	TMR2	Timer2								xxxx xxxx	uuuu uuuu
12h	TMR3L	TMR3 register; low byte								xxxx xxxx	uuuu uuuu
13h	TMR3H	TMR3 register; high byte								xxxx xxxx	uuuu uuuu
14h	PR1	Timer1 period register								xxxx xxxx	uuuu uuuu
15h	PR2	Timer2 period register								xxxx xxxx	uuuu uuuu
16h	PR3L/CA1L	Timer3 period register, low byte/capture1 register; low byte								xxxx xxxx	uuuu uuuu
17h	PR3H/CA1H	Timer3 period register, high byte/capture1 register; high byte								xxxx xxxx	uuuu uuuu
<b>Bank 3</b>											
10h	PW1DCL	DC1	DC0	—	—	—	—	—	—	xx-- ----	uu-- ----
11h	PW2DCL	DC1	DC0	TM2PW2	—	—	—	—	—	xx0- ----	uu0- ----
12h	PW1DCH	DC9	DC8	DC7	DC6	DC5	DC4	DC3	DC2	xxxx xxxx	uuuu uuuu
13h	PW2DCH	DC9	DC8	DC7	DC6	DC5	DC4	DC3	DC2	xxxx xxxx	uuuu uuuu
14h	CA2L	Capture2 low byte								xxxx xxxx	uuuu uuuu
15h	CA2H	Capture2 high byte								xxxx xxxx	uuuu uuuu
16h	TCON1	CA2ED1	CA2ED0	CA1ED1	CA1ED0	T16	TMR3CS	TMR2CS	TMR1CS	0000 0000	0000 0000
17h	TCON2	CA2OVF	CA1OVF	PWM2ON	PWM1ON	CA1/PR3	TMR3ON	TMR2ON	TMR1ON	0000 0000	0000 0000
<b>Unbanked</b>											
18h <sup>(5)</sup>	PRODL	Low Byte of 16-bit Product (8 x 8 Hardware Multiply)								xxxx xxxx	uuuu uuuu
19h <sup>(5)</sup>	PRODH	High Byte of 16-bit Product (8 x 8 Hardware Multiply)								xxxx xxxx	uuuu uuuu

- Legend: x = unknown, u = unchanged, - = unimplemented read as '0', q - value depends on condition. Shaded cells are unimplemented, read as '0'.
- Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for PC<15:8> whose contents are updated from or transferred to the upper byte of the program counter.
- 2: The  $\overline{TO}$  and  $\overline{PD}$  status bits in CPUSTA are not affected by a MCLR reset.
- 3: Other (non power-up) resets include: external reset through MCLR and the Watchdog Timer Reset.
- 4: The following values are for both TBLPTRL and TBLPTRH:  
All PIC17C4X devices (Power-on Reset 0000 0000) and (All other resets 0000 0000) except the PIC17C42 (Power-on Reset xxxx xxxx) and (All other resets uuuu uuuu)
- 5: The PRODL and PRODH registers are not implemented on the PIC17C42.

# PIC17C4X

## 6.2.2.1 ALU STATUS REGISTER (ALUSTA)

The ALUSTA register contains the status bits of the Arithmetic and Logic Unit and the mode control bits for the indirect addressing register.

As with all the other registers, the ALUSTA register can be the destination for any instruction. If the ALUSTA register is the destination for an instruction that affects the Z, DC or C bits, then the write to these three bits is disabled. These bits are set or cleared according to the device logic. Therefore, the result of an instruction with the ALUSTA register as destination may be different than intended.

For example, `CLRF ALUSTA` will clear the upper four bits and set the Z bit. This leaves the ALUSTA register as `0000u1uu` (where `u` = unchanged).

It is recommended, therefore, that only `BCF`, `BSF`, `SWAPF` and `MOVWF` instructions be used to alter the ALUSTA register because these instructions do not affect any status bit. To see how other instructions affect the status bits, see the "Instruction Set Summary."

**Note 1:** The C and DC bits operate as a borrow out bit in subtraction. See the `SUBLW` and `SUBWF` instructions for examples.

**Note 2:** The overflow bit will be set if the 2's complement result exceeds +127 or is less than -128.

Arithmetic and Logic Unit (ALU) is capable of carrying out arithmetic or logical operations on two operands or a single operand. All single operand instructions operate either on the WREG register or a file register. For two operand instructions, one of the operands is the WREG register and the other one is either a file register or an 8-bit immediate constant.

**FIGURE 6-7: ALUSTA REGISTER (ADDRESS: 04h, UNBANKED)**



## 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)**



# PIC17C4X

## 6.2.2.3 TMR0 STATUS/CONTROL REGISTER (T0STA)

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

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

R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	U - 0																				
INTEDG	T0SE	T0CS	PS3	PS2	PS1	PS0	—																				
							bit0																				
<div style="border: 1px solid black; padding: 5px; width: fit-content; margin-left: auto; margin-right: 0;">                     R = Readable bit                      W = Writable bit                      U = Unimplemented, reads as '0'                      -n = Value at POR reset                 </div>																											
<p>bit 7: <b>INTEDG:</b> RA0/INT Pin Interrupt Edge Select bit                      This bit selects the edge upon which the interrupt is detected.                      1 = Rising edge of RA0/INT pin generates interrupt                      0 = Falling edge of RA0/INT pin generates interrupt</p> <p>bit 6: <b>T0SE:</b> Timer0 Clock Input Edge Select bit                      This bit selects the edge upon which TMR0 will increment.  <u>When T0CS = 0</u>                      1 = Rising edge of RA1/T0CKI pin increments TMR0 and/or generates a T0CKIF interrupt                      0 = Falling edge of RA1/T0CKI pin increments TMR0 and/or generates a T0CKIF interrupt  <u>When T0CS = 1</u>                      Don't care</p> <p>bit 5: <b>T0CS:</b> Timer0 Clock Source Select bit                      This bit selects the clock source for Timer0.                      1 = Internal instruction clock cycle (TCY)                      0 = T0CKI pin</p> <p>bit 4-1: <b>PS3:PS0:</b> Timer0 Prescale Selection bits                      These bits select the prescale value for Timer0.</p> <table style="margin-left: 40px;"> <thead> <tr> <th>PS3:PS0</th> <th>Prescale Value</th> </tr> </thead> <tbody> <tr><td>0000</td><td>1:1</td></tr> <tr><td>0001</td><td>1:2</td></tr> <tr><td>0010</td><td>1:4</td></tr> <tr><td>0011</td><td>1:8</td></tr> <tr><td>0100</td><td>1:16</td></tr> <tr><td>0101</td><td>1:32</td></tr> <tr><td>0110</td><td>1:64</td></tr> <tr><td>0111</td><td>1:128</td></tr> <tr><td>1xxx</td><td>1:256</td></tr> </tbody> </table>								PS3:PS0	Prescale Value	0000	1:1	0001	1:2	0010	1:4	0011	1:8	0100	1:16	0101	1:32	0110	1:64	0111	1:128	1xxx	1:256
PS3:PS0	Prescale Value																										
0000	1:1																										
0001	1:2																										
0010	1:4																										
0011	1:8																										
0100	1:16																										
0101	1:32																										
0110	1:64																										
0111	1:128																										
1xxx	1:256																										
<p>bit 0: <b>Unimplemented:</b> Read as '0'</p>																											

## 6.3 Stack Operation

The PIC17C4X devices have a 16 x 16-bit wide hardware stack (Figure 6-1). The stack is not part of either the program or data memory space, and the stack pointer is neither readable nor writable. The PC is “PUSHed” onto the stack when a CALL instruction is executed or an interrupt is acknowledged. The stack is “POPped” in the event of a RETURN, RETLW, or a RETFIE instruction execution. PCLATH is not affected by a “PUSH” or a “POP” operation.

The stack operates as a circular buffer, with the stack pointer initialized to '0' after all resets. There is a stack available bit (STKAV) to allow software to ensure that the stack has not overflowed. The STKAV bit is set after a device reset. When the stack pointer equals Fh, STKAV is cleared. When the stack pointer rolls over from Fh to 0h, the STKAV bit will be held clear until a device reset.

**Note 1:** There is not a status bit for stack underflow. The STKAV bit can be used to detect the underflow which results in the stack pointer being at the top of stack.

**Note 2:** There are no instruction mnemonics called PUSH or POP. These are actions that occur from the execution of the CALL, RETURN, RETLW, and RETFIE instructions, or the vectoring to an interrupt vector.

**Note 3:** After a reset, if a “POP” operation occurs before a “PUSH” operation, the STKAV bit will be cleared. This will appear as if the stack is full (underflow has occurred). If a “PUSH” operation occurs next (before another “POP”), the STKAV bit will be locked clear. Only a device reset will cause this bit to set.

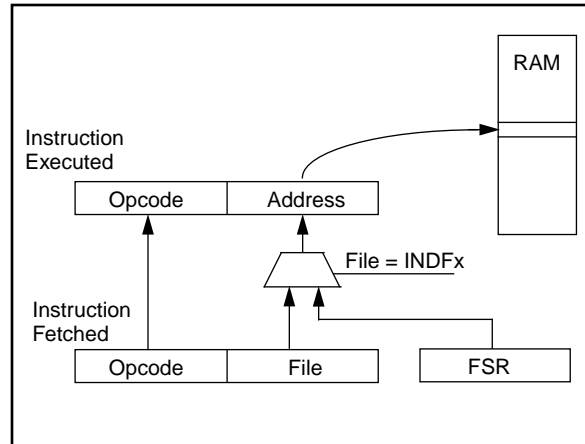
After the device is “PUSHed” sixteen times (without a “POP”), the seventeenth push overwrites the value from the first push. The eighteenth push overwrites the second push (and so on).

## 6.4 Indirect Addressing

Indirect addressing is a mode of addressing data memory where the data memory address in the instruction is not fixed. That is, the register that is to be read or written can be modified by the program. This can be useful for data tables in the data memory. Figure 6-10 shows the operation of indirect addressing. This shows the moving of the value to the data memory address specified by the value of the FSR register.

Example 6-1 shows the use of indirect addressing to clear RAM in a minimum number of instructions. A similar concept could be used to move a defined number of bytes (block) of data to the USART transmit register (TXREG). The starting address of the block of data to be transmitted could easily be modified by the program.

**FIGURE 6-10: INDIRECT ADDRESSING**



# PIC17C4X

## 6.4.1 INDIRECT ADDRESSING REGISTERS

The PIC17C4X has four registers for indirect addressing. These registers are:

- INDF0 and FSR0
- INDF1 and FSR1

Registers INDF0 and INDF1 are not physically implemented. Reading or writing to these registers activates indirect addressing, with the value in the corresponding FSR register being the address of the data. The FSR is an 8-bit register and allows addressing anywhere in the 256-byte data memory address range. For banked memory, the bank of memory accessed is specified by the value in the BSR.

If file INDF0 (or INDF1) itself is read indirectly via an FSR, all '0's are read (Zero bit is set). Similarly, if INDF0 (or INDF1) is written to indirectly, the operation will be equivalent to a NOP, and the status bits are not affected.

## 6.4.2 INDIRECT ADDRESSING OPERATION

The indirect addressing capability has been enhanced over that of the PIC16CXX family. There are two control bits associated with each FSR register. These two bits configure the FSR register to:

- Auto-decrement the value (address) in the FSR after an indirect access
- Auto-increment the value (address) in the FSR after an indirect access
- No change to the value (address) in the FSR after an indirect access

These control bits are located in the ALUSTA register. The FSR1 register is controlled by the FS3:FS2 bits and FSR0 is controlled by the FS1:FS0 bits.

When using the auto-increment or auto-decrement features, the effect on the FSR is not reflected in the ALUSTA register. For example, if the indirect address causes the FSR to equal '0', the Z bit will not be set.

If the FSR register contains a value of 0h, an indirect read will read 0h (Zero bit is set) while an indirect write will be equivalent to a NOP (status bits are not affected).

Indirect addressing allows single cycle data transfers within the entire data space. This is possible with the use of the MOVFPF and MOVFP instructions, where either 'p' or 'f' is specified as INDF0 (or INDF1).

If the source or destination of the indirect address is in banked memory, the location accessed will be determined by the value in the BSR.

A simple program to clear RAM from 20h - FFh is shown in Example 6-1.

### EXAMPLE 6-1: INDIRECT ADDRESSING

```
MOVLW    0x20          ;
MOVWF    FSR0          ; FSR0 = 20h
BCF      ALUSTA, FS1   ; Increment FSR
BSF      ALUSTA, FS0   ; after access
BCF      ALUSTA, C     ; C = 0
MOVLW    END_RAM + 1  ;
LP CLR   INDF0         ; Addr(FSR) = 0
CPFSEQ   FSR0         ; FSR0 = END_RAM+1?
GOTO    LP            ; NO, clear next
:                               ; YES, All RAM is
:                               ; cleared
```

## 6.5 Table Pointer (TBLPTRL and TBLPTRH)

File registers TBLPTRL and TBLPTRH form a 16-bit pointer to address the 64K program memory space. The table pointer is used by instructions `TABLWT` and `TABLRD`.

The `TABLRD` and the `TABLWT` instructions allow transfer of data between program and data space. The table pointer serves as the 16-bit address of the data word within the program memory. For a more complete description of these registers and the operation of Table Reads and Table Writes, see Section 7.0.

## 6.6 Table Latch (TBLATH, TBLATL)

The table latch (TBLAT) is a 16-bit register, with TBLATH and TBLATL referring to the high and low bytes of the register. It is not mapped into data or program memory. The table latch is used as a temporary holding latch during data transfer between program and data memory (see descriptions of instructions `TABLRD`, `TABLWT`, `TLRD` and `TLWT`). For a more complete description of these registers and the operation of Table Reads and Table Writes, see Section 7.0.



## 6.7 Program Counter Module

The Program Counter (PC) is a 16-bit register. PCL, the low byte of the PC, is mapped in the data memory. PCL is readable and writable just as is any other register. PCH is the high byte of the PC and is not directly addressable. Since PCH is not mapped in data or program memory, an 8-bit register PCLATH (PC high latch) is used as a holding latch for the high byte of the PC. PCLATH is mapped into data memory. The user can read or write PCH through PCLATH.

The 16-bit wide PC is incremented after each instruction fetch during Q1 unless:

- Modified by GOTO, CALL, LCALL, RETURN, RETLW, or RETFIE instruction
- Modified by an interrupt response
- Due to destination write to PCL by an instruction

“Skips” are equivalent to a forced NOP cycle at the skipped address.

Figure 6-11 and Figure 6-12 show the operation of the program counter for various situations.

**FIGURE 6-11: PROGRAM COUNTER OPERATION**



**FIGURE 6-12: PROGRAM COUNTER USING THE CALL AND GOTO INSTRUCTIONS**



Using Figure 6-11, the operations of the PC and PCLATH for different instructions are as follows:

- LCALL instructions:**  
An 8-bit destination address is provided in the instruction (opcode). PCLATH is unchanged.  
PCLATH → PCH  
Opcode<7:0> → PCL
- Read instructions on PCL:**  
Any instruction that reads PCL.  
PCL → data bus → ALU or destination  
PCH → PCLATH
- Write instructions on PCL:**  
Any instruction that writes to PCL.  
8-bit data → data bus → PCL  
PCLATH → PCH
- Read-Modify-Write instructions on PCL:**  
Any instruction that does a read-write-modify operation on PCL, such as ADDWF PCL.  
Read: PCL → data bus → ALU  
Write: 8-bit result → data bus → PCL  
PCLATH → PCH
- RETURN instruction:**  
PCH → PCLATH  
Stack<MRU> → PC<15:0>

Using Figure 6-12, the operation of the PC and PCLATH for GOTO and CALL instructions is as follows:

**CALL, GOTO instructions:**

A 13-bit destination address is provided in the instruction (opcode).

Opcode<12:0> → PC <12:0>

PC<15:13> → PCLATH<7:5>

Opcode<12:8> → PCLATH <4:0>

The read-modify-write only affects the PCL with the result. PCH is loaded with the value in the PCLATH. For example, ADDWF PCL will result in a jump within the current page. If PC = 03F0h, WREG = 30h and PCLATH = 03h before instruction, PC = 0320h after the instruction. To accomplish a true 16-bit computed jump, the user needs to compute the 16-bit destination address, write the high byte to PCLATH and then write the low value to PCL.

The following PC related operations do not change PCLATH:

- LCALL, RETLW, and RETFIE instructions.
- Interrupt vector is forced onto the PC.
- Read-modify-write instructions on PCL (e.g. BSF PCL).

# PIC17C4X

## 6.8 Bank Select Register (BSR)

The BSR is used to switch between banks in the data memory area (Figure 6-13). In the PIC17C42, PIC17CR42, and PIC17C42A only the lower nibble is implemented. While in the PIC17C43, PIC17CR43, and PIC17C44 devices, the entire byte is implemented. The lower nibble is used to select the peripheral register bank. The upper nibble is used to select the general purpose memory bank.

All the Special Function Registers (SFRs) are mapped into the data memory space. In order to accommodate the large number of registers, a banking scheme has been used. A segment of the SFRs, from address 10h to address 17h, is banked. The lower nibble of the bank select register (BSR) selects the currently active "peripheral bank." Effort has been made to group the peripheral registers of related functionality in one bank. However, it will still be necessary to switch from bank to bank in order to address all peripherals related to a single task. To assist this, a `MOVLB` bank instruction is in the instruction set.

For the PIC17C43, PIC17CR43, and PIC17C44 devices, the need for a large general purpose memory space dictated a general purpose RAM banking scheme. The upper nibble of the BSR selects the currently active general purpose RAM bank. To assist this, a `MOVLR` bank instruction has been provided in the instruction set.

If the currently selected bank is not implemented (such as Bank 13), any read will read all '0's. Any write is completed to the bit bucket and the ALU status bits will be set/cleared as appropriate.

**Note:** Registers in Bank 15 in the Special Function Register area, are reserved for Microchip use. Reading of registers in this bank may cause random values to be read.

FIGURE 6-13: BSR OPERATION (PIC17C43/R43/44)



## 7.0 TABLE READS AND TABLE WRITES

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

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

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

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

**FIGURE 7-1: TLWT INSTRUCTION OPERATION**



**FIGURE 7-2: TABLWT INSTRUCTION OPERATION**



**FIGURE 7-3: TLRD INSTRUCTION OPERATION**



**FIGURE 7-4: TABLRD INSTRUCTION OPERATION**



## 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).

## 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

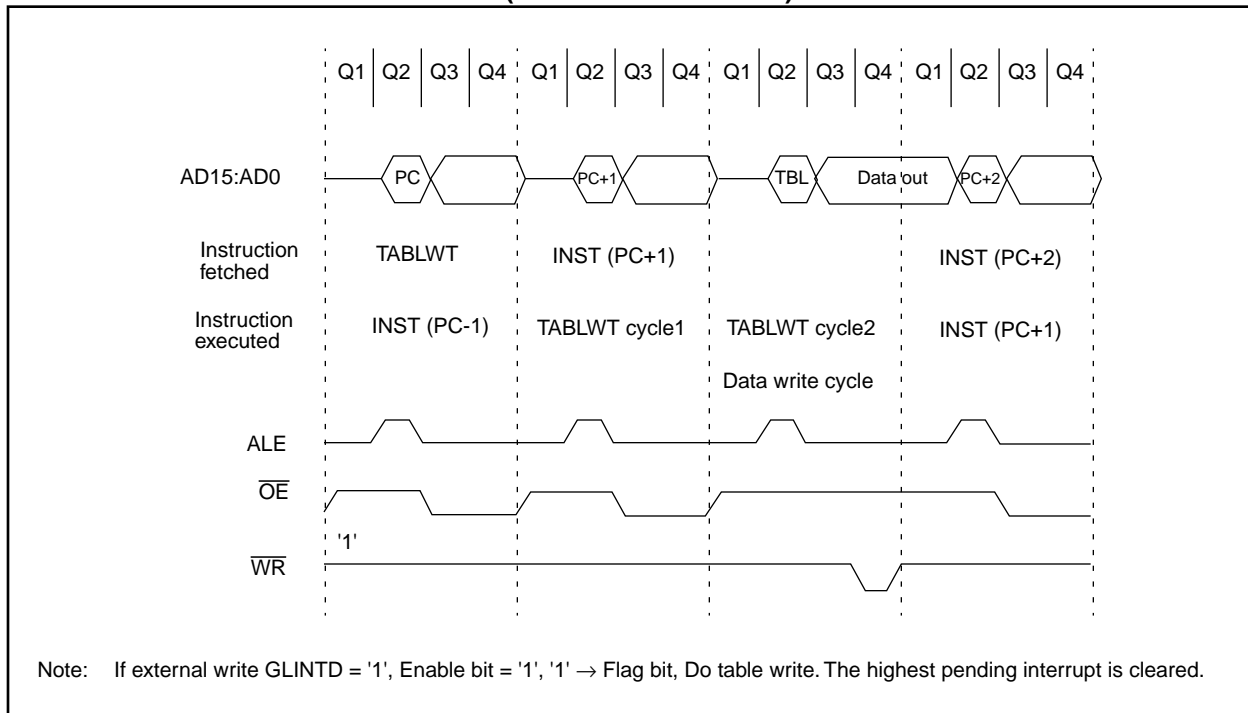
The “i” operand of the `TABLWT` instruction can specify that the value in the 16-bit `TBLPTR` register is automatically incremented for the next write. In Example 7-1, the `TBLPTR` register is not automatically incremented.

#### EXAMPLE 7-1: TABLE WRITE

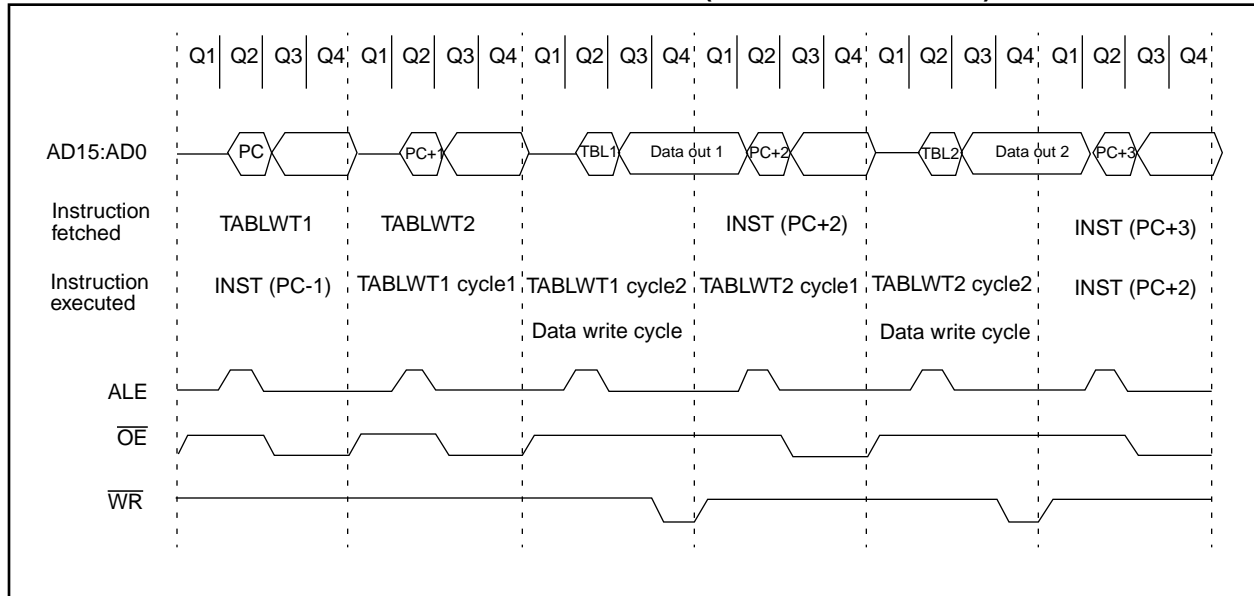
```

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)**



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



## 7.3 Table Reads

The table read allows the program memory to be read. This allows constant data to be stored in the program memory space, and retrieved into data memory when needed. Example 7-2 reads the 16-bit value at program memory address TBLPTR. After the dummy byte has been read from the TABLATH, the TABLATH is loaded with the 16-bit data from program memory address TBLPTR + 1. The first read loads the data into the latch, and can be considered a dummy read (unknown data loaded into 'f'). INDF0 should be configured for either auto-increment or auto-decrement.

### EXAMPLE 7-2: TABLE READ

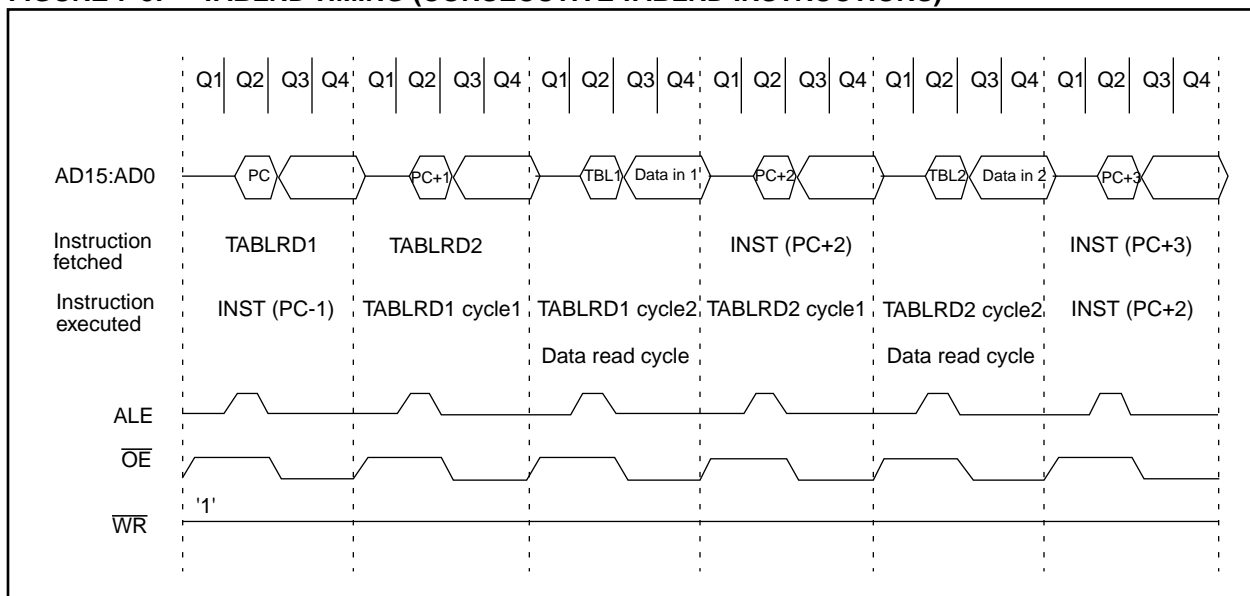
```

MOVLW  HIGH (TBL_ADDR) ; Load the Table
MOVWF  TBLPTRH         ; address
MOVLW  LOW  (TBL_ADDR) ;
MOVWF  TBLPTRL         ;
TABLRD 0,0,DUMMY      ; Dummy read,
                          ; Updates TABLATCH
TLRD   1, INDF0        ; Read HI byte
                          ; of TABLATCH
TABLRD 0,1,INDF0      ; Read LO byte
                          ; of TABLATCH and
                          ; Update TABLATCH
    
```

**FIGURE 7-7: TABLRD TIMING**



**FIGURE 7-8: TABLRD TIMING (CONSECUTIVE TABLRD INSTRUCTIONS)**





## 8.0 HARDWARE MULTIPLIER

All PIC17C4X devices except the PIC17C42, have an 8 x 8 hardware multiplier included in the ALU of the device. By making the multiply a hardware operation, it completes in a single instruction cycle. This is an unsigned multiply that gives a 16-bit result. The result is stored into the 16-bit PRODUct register (PRODH:PRODL). The multiplier does not affect any flags in the ALUSTA register.

Making the 8 x 8 multiplier execute in a single cycle gives the following advantages:

- Higher computational throughput
- Reduces code size requirements for multiply algorithms

The performance increase allows the device to be used in applications previously reserved for Digital Signal Processors.

Table 8-1 shows a performance comparison between the PIC17C42 and all other PIC17CXX devices, which have the single cycle hardware multiply.

Example 8-1 shows the sequence to do an 8 x 8 unsigned multiply. Only one instruction is required when one argument of the multiply is already loaded in the WREG register.

Example 8-2 shows the sequence to do an 8 x 8 signed multiply. To account for the sign bits of the arguments, each argument's most significant bit (MSb) is tested and the appropriate subtractions are done.

### EXAMPLE 8-1: 8 x 8 MULTIPLY ROUTINE

```
MOVFP   ARG1, WREG
MULWF   ARG2           ; ARG1 * ARG2 ->
                        ;   PRODH:PRODL
```

### EXAMPLE 8-2: 8 x 8 SIGNED MULTIPLY ROUTINE

```
MOVFP   ARG1, WREG
MULWF   ARG2           ; ARG1 * ARG2 ->
                        ;   PRODH:PRODL

BTFSC   ARG2, SB      ; Test Sign Bit
SUBWF   PRODH, F      ; PRODH = PRODH
                        ;   - ARG1

MOVFP   ARG2, WREG
BTFSC   ARG1, SB      ; Test Sign Bit
SUBWF   PRODH, F      ; PRODH = PRODH
                        ;   - ARG2
```

**TABLE 8-1: PERFORMANCE COMPARISON**

Routine	Device	Program Memory (Words)	Cycles (Max)	Time	
				@ 25 MHz	@ 33 MHz
8 x 8 unsigned	PIC17C42	13	69	11.04 $\mu$ s	N/A
	All other PIC17CXX devices	1	1	160 ns	121 ns
8 x 8 signed	PIC17C42	—	—	—	N/A
	All other PIC17CXX devices	6	6	960 ns	727 ns
16 x 16 unsigned	PIC17C42	21	242	38.72 $\mu$ s	N/A
	All other PIC17CXX devices	24	24	3.84 $\mu$ s	2.91 $\mu$ s
16 x 16 signed	PIC17C42	52	254	40.64 $\mu$ s	N/A
	All other PIC17CXX devices	36	36	5.76 $\mu$ s	4.36 $\mu$ s

# PIC17C4X

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

$$\begin{aligned} \text{RES3:RES0} &= \text{ARG1H:ARG1L} * \text{ARG2H:ARG2L} \\ &= (\text{ARG1H} * \text{ARG2H} * 2^{16}) + \\ &\quad (\text{ARG1H} * \text{ARG2L} * 2^8) + \\ &\quad (\text{ARG1L} * \text{ARG2H} * 2^8) + \\ &\quad (\text{ARG1L} * \text{ARG2L}) \end{aligned}$$

## EXAMPLE 8-3: 16 x 16 MULTIPLY ROUTINE

```
MOVFP ARG1L, WREG
MULWF ARG2L      ; ARG1L * ARG2L ->
                  ;   PRODH:PRODL
MOVFP PRODH, RES1 ;
MOVFP PRODL, RES0 ;
;
MOVFP ARG1H, WREG
MULWF ARG2H      ; ARG1H * ARG2H ->
                  ;   PRODH:PRODL
MOVFP PRODH, RES3 ;
MOVFP 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

$$\begin{aligned}
 & \text{RES3:RES0} \\
 & = \text{ARG1H:ARG1L} * \text{ARG2H:ARG2L} \\
 & = (\text{ARG1H} * \text{ARG2H} * 2^{16}) \quad + \\
 & \quad (\text{ARG1H} * \text{ARG2L} * 2^8) \quad + \\
 & \quad (\text{ARG1L} * \text{ARG2H} * 2^8) \quad + \\
 & \quad (\text{ARG1L} * \text{ARG2L}) \quad + \\
 & \quad (-1 * \text{ARG2H} <7> * \text{ARG1H:ARG1L} * 2^{16}) \quad + \\
 & \quad (-1 * \text{ARG1H} <7> * \text{ARG2H:ARG2L} * 2^{16})
 \end{aligned}$$

## EXAMPLE 8-4: 16 x 16 SIGNED MULTIPLY ROUTINE

```

MOVFP ARG1L, WREG
MULWF ARG2L      ; ARG1L * ARG2L ->
                  ;   PRODH:PRODL

MOVFP PRODH, RES1 ;
MOVFP PRODL, RES0 ;

;

MOVFP ARG1H, WREG
MULWF ARG2H      ; ARG1H * ARG2H ->
                  ;   PRODH:PRODL

MOVFP PRODH, RES3 ;
MOVFP 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  ;
CLRWF 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  ;
CLRWF 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
:

```

# PIC17C4X

---

NOTES:

## 9.0 I/O PORTS

The PIC17C4X devices have five I/O ports, PORTA through PORTE. PORTB through PORTE have a corresponding Data Direction Register (DDR), which is used to configure the port pins as inputs or outputs. These five ports are made up of 33 I/O pins. Some of these ports pins are multiplexed with alternate functions.

PORTC, PORTD, and PORTE are multiplexed with the system bus. These pins are configured as the system bus when the device's configuration bits are selected to Microprocessor or Extended Microcontroller modes. In the two other microcontroller modes, these pins are general purpose I/O.

PORTA and PORTB are multiplexed with the peripheral features of the device. These peripheral features are:

- Timer modules
- Capture module
- PWM module
- USART/SCI module
- External Interrupt pin

When some of these peripheral modules are turned on, the port pin will automatically configure to the alternate function. The modules that do this are:

- PWM module
- USART/SCI module

When a pin is automatically configured as an output by a peripheral module, the pins data direction (DDR) bit is unknown. After disabling the peripheral module, the user should re-initialize the DDR bit to the desired configuration.

The other peripheral modules (which require an input) must have their data direction bit configured appropriately.

**Note:** A pin that is a peripheral input, can be configured as an output (DDR<sub>x<y></sub> is cleared). The peripheral events will be determined by the action output on the port pin.

## 9.1 PORTA Register

PORTA is a 6-bit wide latch. PORTA does not have a corresponding Data Direction Register (DDR).

Reading PORTA reads the status of the pins.

The RA1 pin is multiplexed with TMR0 clock input, and RA4 and RA5 are multiplexed with the USART functions. The control of RA4 and RA5 as outputs is automatically configured by the USART module.

### 9.1.1 USING RA2, RA3 AS OUTPUTS

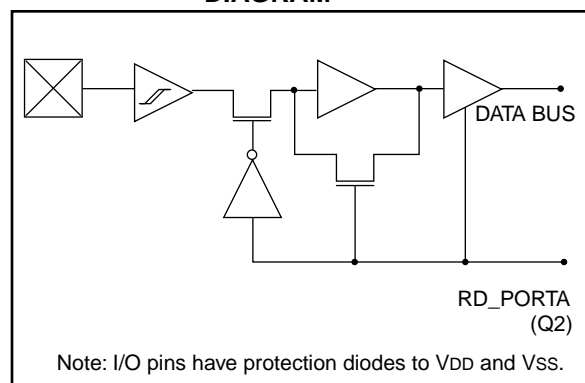
The RA2 and RA3 pins are open drain outputs. To use the RA2 or the RA3 pin(s) as output(s), simply write to the PORTA register the desired value. A '0' will cause the pin to drive low, while a '1' will cause the pin to float (hi-impedance). An external pull-up resistor should be used to pull the pin high. Writes to PORTA will not affect the other pins.

**Note:** When using the RA2 or RA3 pin(s) as output(s), read-modify-write instructions (such as BCF, BSF, BTG) on PORTA are not recommended.

Such operations read the port pins, do the desired operation, and then write this value to the data latch. This may inadvertently cause the RA2 or RA3 pins to switch from input to output (or vice-versa).

It is recommended to use a shadow register for PORTA. Do the bit operations on this shadow register and then move it to PORTA.

**FIGURE 9-1: RA0 AND RA1 BLOCK DIAGRAM**



# PIC17C4X

**FIGURE 9-2: RA2 AND RA3 BLOCK DIAGRAM**



**FIGURE 9-3: RA4 AND RA5 BLOCK DIAGRAM**



**TABLE 9-1: PORTA FUNCTIONS**

Name	Bit0	Buffer Type	Function
RA0/INT	bit0	ST	Input or external interrupt input.
RA1/T0CKI	bit1	ST	Input or clock input to the TMR0 timer/counter, and/or an external interrupt input.
RA2	bit2	ST	Input/Output. Output is open drain type.
RA3	bit3	ST	Input/Output. Output is open drain type.
RA4/RX/DT	bit4	ST	Input or USART Asynchronous Receive or USART Synchronous Data.
RA5/TX/CK	bit5	ST	Input or USART Asynchronous Transmit or USART Synchronous Clock.
$\overline{\text{RBP}}\text{U}$	bit7	—	Control bit for PORTB weak pull-ups.

Legend: ST = Schmitt Trigger input.

**TABLE 9-2: REGISTERS/BITS ASSOCIATED WITH PORTA**

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)
10h, Bank 0	PORTA	$\overline{\text{RBP}}\text{U}$	—	RA5	RA4	RA3	RA2	RA1/T0CKI	RA0/INT	0-xx xxxx	0-uu uuuu
05h, Unbanked	T0STA	INTEDG	T0SE	T0CS	PS3	PS2	PS1	PS0	—	0000 000-	0000 000-
13h, Bank 0	RCSTA	SPEN	RC9	SREN	CREN	—	FERR	OERR	RC9D	0000 -00x	0000 -00u
15h, Bank 0	TXSTA	CSRC	TX9	TXEN	SYNC	—	—	TRMT	TX9D	0000 --1x	0000 --1u

Legend: x = unknown, u = unchanged, - = unimplemented reads as '0'. Shaded cells are not used by PORTA.

Note 1: Other (non power-up) resets include: external reset through MCLR and the Watchdog Timer Reset.

## 9.2 PORTB and DDRB Registers

PORTB is an 8-bit wide bi-directional port. The corresponding data direction register is DDRB. A '1' in DDRB configures the corresponding port pin as an input. A '0' in the DDRB register configures the corresponding port pin as an output. Reading PORTB reads the status of the pins, whereas writing to it will write to the port latch.

Each of the PORTB pins has a weak internal pull-up. A single control bit can turn on all the pull-ups. This is done by clearing the  $\overline{\text{RBP}}\text{U}$  (PORTA<7>) bit. The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are enabled on any reset.

PORTB also has an interrupt on change feature. Only pins configured as inputs can cause this interrupt to occur (i.e. any RB7:RB0 pin configured as an output is excluded from the interrupt on change comparison). The input pins (of RB7:RB0) are compared with the value in the PORTB data latch. The "mismatch" outputs of RB7:RB0 are OR'ed together to generate the PORTB Interrupt Flag RBIF (PIR<7>).

This interrupt can wake the device from SLEEP. The user, in the interrupt service routine, can clear the interrupt by:

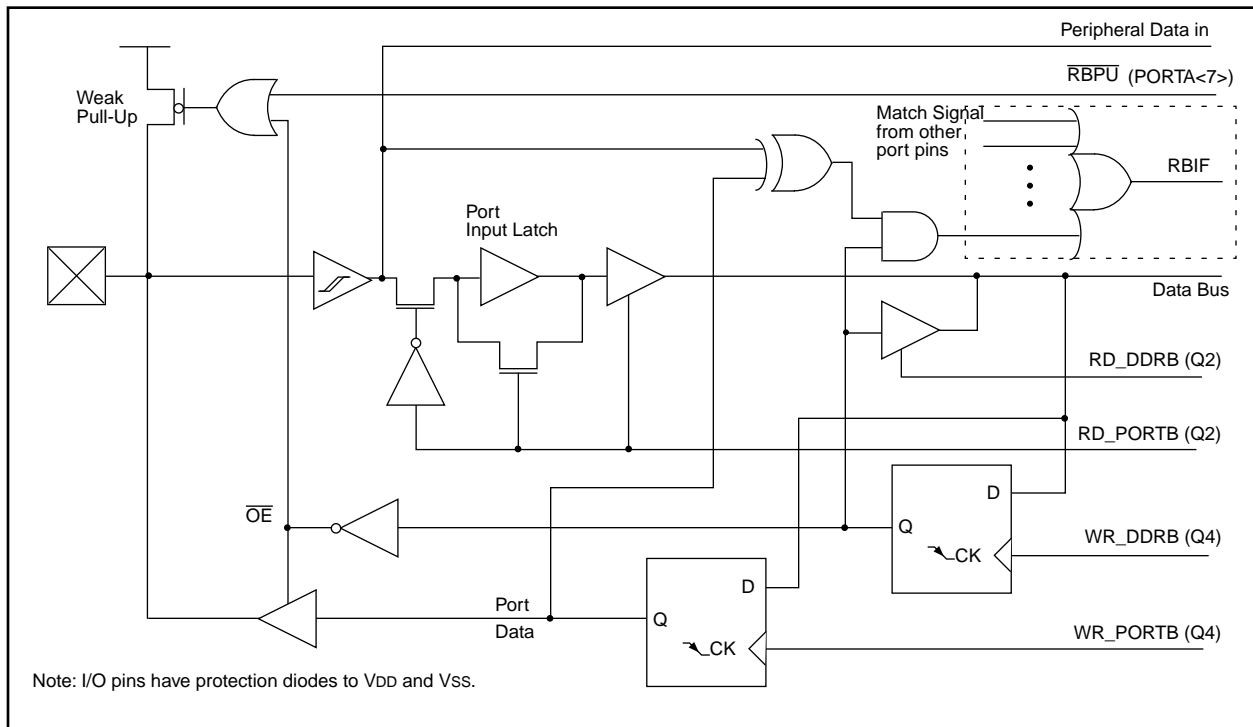
- a) Read-Write PORTB (such as; `MOVVPF PORTB, PORTB`). This will end mismatch condition.
- b) Then, clear the RBIF bit.

A mismatch condition will continue to set the RBIF bit. Reading then writing PORTB will end the mismatch condition, and allow the RBIF bit to be cleared.

This interrupt on mismatch feature, together with software configurable pull-ups on this port, allows easy interface to a key pad and make it possible for wake-up on key-depression. For an example, refer to AN552 in the *Embedded Control Handbook*.

The interrupt on change feature is recommended for wake-up on operations where PORTB is only used for the interrupt on change feature and key depression operation.

**FIGURE 9-4: BLOCK DIAGRAM OF RB<7:4> AND RB<1:0> PORT PINS**



# PIC17C4X

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





Example 9-1 shows the instruction sequence to initialize PORTB. The Bank Select Register (BSR) must be selected to Bank 0 for the port to be initialized.

## EXAMPLE 9-1: INITIALIZING PORTB

```

MOVLB 0           ; Select Bank 0
CLRF  PORTB      ; Initialize PORTB by clearing
                  ; output data latches
MOVLW 0xCF       ; Value used to initialize
                  ; data direction
MOVWF  DDRB      ; Set RB<3:0> as inputs
                  ; RB<5:4> as outputs
                  ; RB<7:6> as inputs
    
```

**TABLE 9-3: PORTB FUNCTIONS**

Name	Bit	Buffer Type	Function
RB0/CAP1	bit0	ST	Input/Output or the RB0/CAP1 input pin. Software programmable weak pull-up and interrupt on change features.
RB1/CAP2	bit1	ST	Input/Output or the RB1/CAP2 input pin. Software programmable weak pull-up and interrupt on change features.
RB2/PWM1	bit2	ST	Input/Output or the RB2/PWM1 output pin. Software programmable weak pull-up and interrupt on change features.
RB3/PWM2	bit3	ST	Input/Output or the RB3/PWM2 output pin. Software programmable weak pull-up and interrupt on change features.
RB4/TCLK12	bit4	ST	Input/Output or the external clock input to Timer1 and Timer2. Software programmable weak pull-up and interrupt on change features.
RB5/TCLK3	bit5	ST	Input/Output or the external clock input to Timer3. Software programmable weak pull-up and interrupt on change features.
RB6	bit6	ST	Input/Output pin. Software programmable weak pull-up and interrupt on change features.
RB7	bit7	ST	Input/Output pin. Software programmable weak pull-up and interrupt on change features.

Legend: ST = Schmitt Trigger input.

**TABLE 9-4: REGISTERS/BITS ASSOCIATED WITH PORTB**

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)
12h, Bank 0	PORTB	PORTB data latch								xxxx xxxx	uuuu uuuu
11h, Bank 0	DDRB	Data direction register for PORTB								1111 1111	1111 1111
10h, Bank 0	PORTA	RBPU	—	RA5	RA4	RA3	RA2	RA1/T0CKI	RA0/INT	0-xx xxxx	0-uu uuuu
06h, Unbanked	CPUSTA	—	—	STKAV	GLINTD	T0	PD	—	—	--11 11--	--11 qq--
07h, Unbanked	INTSTA	PEIF	T0CKIF	T0IF	INTF	PEIE	T0CKIE	T0IE	INTE	0000 0000	0000 0000
16h, Bank 1	PIR	RBIF	TMR3IF	TMR2IF	TMR1IF	CA2IF	CA1IF	TXIF	RCIF	0000 0010	0000 0010
17h, Bank 1	PIE	RBIE	TMR3IE	TMR2IE	TMR1IE	CA2IE	CA1IE	TXIE	RCIE	0000 0000	0000 0000
16h, Bank 3	TCON1	CA2ED1	CA2ED0	CA1ED1	CA1ED0	T16	TMR3CS	TMR2CS	TMR1CS	0000 0000	0000 0000
17h, Bank 3	TCON2	CA2OVF	CA1OVF	PWM2ON	PWM1ON	CA1/PR3	TMR3ON	TMR2ON	TMR1ON	0000 0000	0000 0000

Legend: x = unknown, u = unchanged, - = unimplemented read as '0', q = Value depends on condition.

Shaded cells are not used by PORTB.

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

# PIC17C4X

## 9.3 PORTC and DDRC Registers

PORTC is an 8-bit bi-directional port. The corresponding data direction register is DDRC. A '1' in DDRC configures the corresponding port pin as an input. A '0' in the DDRC register configures the corresponding port pin as an output. Reading PORTC reads the status of the pins, whereas writing to it will write to the port latch. PORTC is multiplexed with the system bus. When operating as the system bus, PORTC is the low order byte of the address/data bus (AD7:AD0). The timing for the system bus is shown in the Electrical Characteristics section.

**Note:** This port is configured as the system bus when the device's configuration bits are selected to Microprocessor or Extended Microcontroller modes. In the two other microcontroller modes, this port is a general purpose I/O.

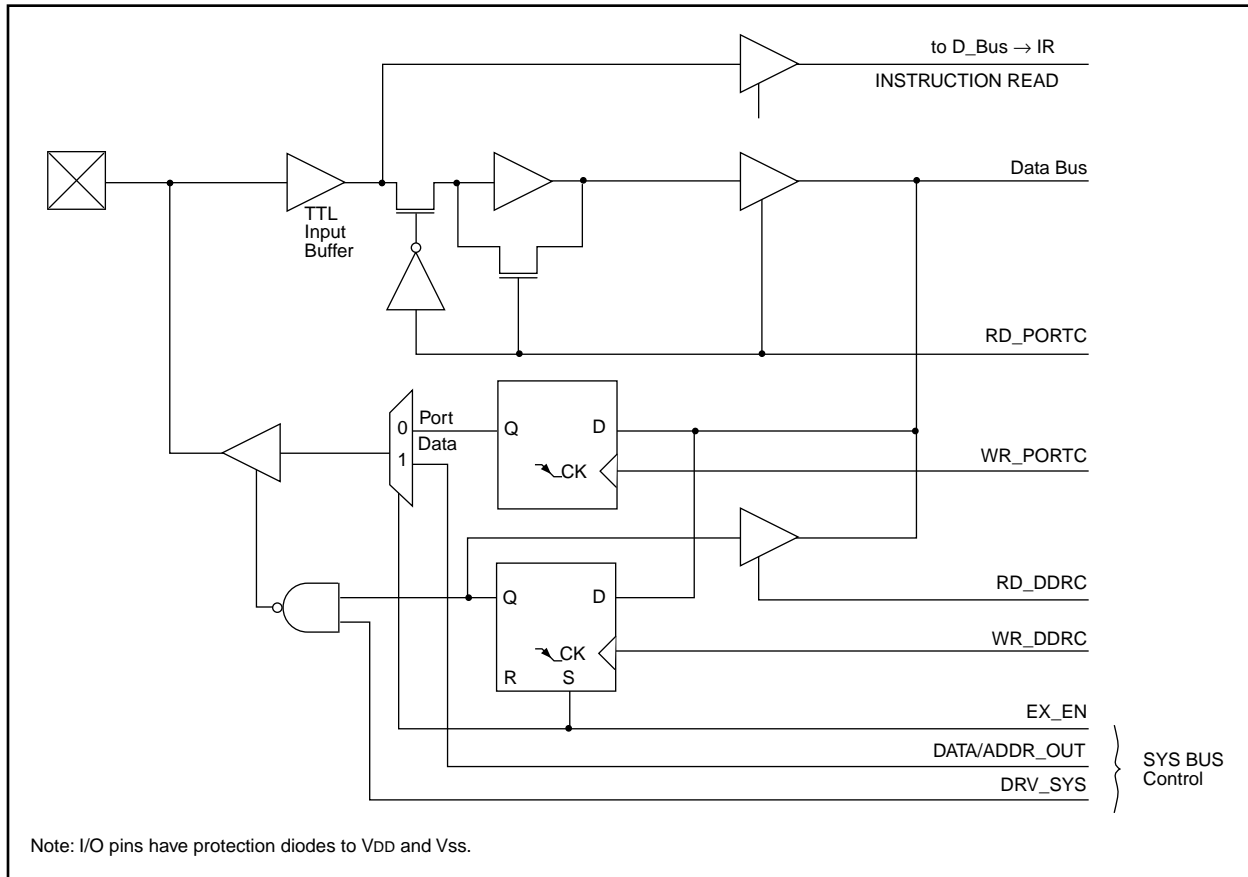
Example 9-2 shows the instruction sequence to initialize PORTC. The Bank Select Register (BSR) must be selected to Bank 1 for the port to be initialized.

### EXAMPLE 9-2: INITIALIZING PORTC

```

MOVLB 1           ; Select Bank 1
CLRWF PORTC      ; Initialize PORTC data
                  ; latches before setting
                  ; the data direction
                  ; register
MOVLW 0xCF       ; Value used to initialize
                  ; data direction
MOVWF DDRC       ; Set RC<3:0> as inputs
                  ; RC<5:4> as outputs
                  ; RC<7:6> as inputs
    
```

FIGURE 9-6: BLOCK DIAGRAM OF RC<7:0> PORT PINS



**TABLE 9-5: PORTC FUNCTIONS**

Name	Bit	Buffer Type	Function
RC0/AD0	bit0	TTL	Input/Output or system bus address/data pin.
RC1/AD1	bit1	TTL	Input/Output or system bus address/data pin.
RC2/AD2	bit2	TTL	Input/Output or system bus address/data pin.
RC3/AD3	bit3	TTL	Input/Output or system bus address/data pin.
RC4/AD4	bit4	TTL	Input/Output or system bus address/data pin.
RC5/AD5	bit5	TTL	Input/Output or system bus address/data pin.
RC6/AD6	bit6	TTL	Input/Output or system bus address/data pin.
RC7/AD7	bit7	TTL	Input/Output or system bus address/data pin.

Legend: TTL = TTL input.

**TABLE 9-6: REGISTERS/BITS ASSOCIATED WITH PORTC**

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)
11h, Bank 1	PORTC	RC7/ AD7	RC6/ AD6	RC5/ AD5	RC4/ AD4	RC3/ AD3	RC2/ AD2	RC1/ AD1	RC0/ AD0	xxxx xxxx	uuuu uuuu
10h, Bank 1	DDRC	Data direction register for PORTC								1111 1111	1111 1111

Legend: x = unknown, u = unchanged.

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

# PIC17C4X

## 9.4 PORTD and DDRD Registers

PORTD is an 8-bit bi-directional port. The corresponding data direction register is DDRD. A '1' in DDRD configures the corresponding port pin as an input. A '0' in the DDRD register configures the corresponding port pin as an output. Reading PORTD reads the status of the pins, whereas writing to it will write to the port latch. PORTD is multiplexed with the system bus. When operating as the system bus, PORTD is the high order byte of the address/data bus (AD15:AD8). The timing for the system bus is shown in the Electrical Characteristics section.

**Note:** This port is configured as the system bus when the device's configuration bits are selected to Microprocessor or Extended Microcontroller modes. In the two other microcontroller modes, this port is a general purpose I/O.

Example 9-3 shows the instruction sequence to initialize PORTD. The Bank Select Register (BSR) must be selected to Bank 1 for the port to be initialized.

### EXAMPLE 9-3: INITIALIZING PORTD

```

MOVLB 1           ; Select Bank 1
CLRF  PORTD      ; Initialize PORTD data
                  ; latches before setting
                  ; the data direction
                  ; register
MOVWL 0xCF       ; Value used to initialize
                  ; data direction
MOVWF DDRD       ; Set RD<3:0> as inputs
                  ; RD<5:4> as outputs
                  ; RD<7:6> as inputs
    
```

**FIGURE 9-7: PORTD BLOCK DIAGRAM (IN I/O PORT MODE)**



**TABLE 9-7: PORTD FUNCTIONS**

Name	Bit	Buffer Type	Function
RD0/AD8	bit0	TTL	Input/Output or system bus address/data pin.
RD1/AD9	bit1	TTL	Input/Output or system bus address/data pin.
RD2/AD10	bit2	TTL	Input/Output or system bus address/data pin.
RD3/AD11	bit3	TTL	Input/Output or system bus address/data pin.
RD4/AD12	bit4	TTL	Input/Output or system bus address/data pin.
RD5/AD13	bit5	TTL	Input/Output or system bus address/data pin.
RD6/AD14	bit6	TTL	Input/Output or system bus address/data pin.
RD7/AD15	bit7	TTL	Input/Output or system bus address/data pin.

Legend: TTL = TTL input.

**TABLE 9-8: REGISTERS/BITS ASSOCIATED WITH PORTD**

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)
13h, Bank 1	PORTD	RD7/ AD15	RD6/ AD14	RD5/ AD13	RD4/ AD12	RD3/ AD11	RD2/ AD10	RD1/ AD9	RD0/ AD8	xxxx xxxx	uuuu uuuu
12h, Bank 1	DDRD	Data direction register for PORTD								1111 1111	1111 1111

Legend: x = unknown, u = unchanged.

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

# PIC17C4X

## 9.4.1 PORTE AND DDRE REGISTER

PORTE is a 3-bit bi-directional port. The corresponding data direction register is DDRE. A '1' in DDRE configures the corresponding port pin as an input. A '0' in the DDRE register configures the corresponding port pin as an output. Reading PORTE reads the status of the pins, whereas writing to it will write to the port latch. PORTE is multiplexed with the system bus. When operating as the system bus, PORTE contains the control signals for the address/data bus (AD15:AD0). These control signals are Address Latch Enable (ALE), Output Enable ( $\overline{OE}$ ), and Write ( $\overline{WR}$ ). The control signals  $\overline{OE}$  and  $\overline{WR}$  are active low signals. The timing for the system bus is shown in the Electrical Characteristics section.

**Note:** This port is configured as the system bus when the device's configuration bits are selected to Microprocessor or Extended Microcontroller modes. In the two other microcontroller modes, this port is a general purpose I/O.

Example 9-4 shows the instruction sequence to initialize PORTE. The Bank Select Register (BSR) must be selected to Bank 1 for the port to be initialized.

### EXAMPLE 9-4: INITIALIZING PORTE

```

MOVLB 1           ; Select Bank 1
CLRFB PORTE      ; Initialize PORTE data
                  ; latches before setting
                  ; the data direction
                  ; register
MOVWL 0x03       ; Value used to initialize
                  ; data direction
MOVWF DDRE       ; Set RE<1:0> as inputs
                  ; RE<2> as outputs
                  ; RE<7:3> are always
                  ; read as '0'
    
```

**FIGURE 9-8: PORTE BLOCK DIAGRAM (IN I/O PORT MODE)**



**TABLE 9-9: PORTE FUNCTIONS**

Name	Bit	Buffer Type	Function
RE0/ALE	bit0	TTL	Input/Output or system bus Address Latch Enable (ALE) control pin.
RE1/ $\overline{OE}$	bit1	TTL	Input/Output or system bus Output Enable ( $\overline{OE}$ ) control pin.
RE2/ $\overline{WR}$	bit2	TTL	Input/Output or system bus Write ( $\overline{WR}$ ) control pin.

Legend: TTL = TTL input.

**TABLE 9-10: REGISTERS/BITS ASSOCIATED WITH PORTE**

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)
15h, Bank 1	PORTE	—	—	—	—	—	RE2/ $\overline{WR}$	RE1/ $\overline{OE}$	RE0/ALE	---- -xxx	---- -uuu
14h, Bank 1	DDRE	Data direction register for PORTE								---- -111	---- -111

Legend: x = unknown, u = unchanged, - = unimplemented read as '0'. Shaded cells are not used by PORTE.

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

## 9.5 I/O Programming Considerations

### 9.5.1 BI-DIRECTIONAL I/O PORTS

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

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

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

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

```

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

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

### 9.5.2 SUCCESSIVE OPERATIONS ON I/O PORTS

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

**FIGURE 9-9: SUCCESSIVE I/O OPERATION**





## 10.0 OVERVIEW OF TIMER RESOURCES

The PIC17C4X has four timer modules. Each module can generate an interrupt to indicate that an event has occurred. These timers are called:

- Timer0 - 16-bit timer with programmable 8-bit prescaler
- Timer1 - 8-bit timer
- Timer2 - 8-bit timer
- Timer3 - 16-bit timer

For enhanced time-base functionality, two input Captures and two Pulse Width Modulation (PWM) outputs are possible. The PWMs use the TMR1 and TMR2 resources and the input Captures use the TMR3 resource.

### 10.1 Timer0 Overview

The Timer0 module is a simple 16-bit overflow counter. The clock source can be either the internal system clock ( $F_{osc}/4$ ) or an external clock.

The Timer0 module also has a programmable prescaler option. The PS3:PS0 bits (T0STA<4:1>) determine the prescaler value. TMR0 can increment at the following rates: 1:1, 1:2, 1:4, 1:8, 1:16, 1:32, 1:64, 1:128, 1:256.

When Timer0's clock source is an external clock, the Timer0 module can be selected to increment on either the rising or falling edge.

Synchronization of the external clock occurs after the prescaler. When the prescaler is used, the external clock frequency may be higher than the device's frequency. The maximum frequency is 50 MHz, given the high and low time requirements of the clock.

### 10.2 Timer1 Overview

The Timer1 module is an 8-bit timer/counter with an 8-bit period register (PR1). When the TMR1 value rolls over from the period match value to 0h, the TMR1IF flag is set, and an interrupt will be generated when enabled. In counter mode, the clock comes from the RB4/TCLK12 pin, which can also be selected to be the clock for the Timer2 module.

TMR1 can be concatenated to TMR2 to form a 16-bit timer. The TMR1 register is the LSB and TMR2 is the MSB. When in the 16-bit timer mode, there is a corresponding 16-bit period register (PR2:PR1). When the TMR2:TMR1 value rolls over from the period match value to 0h, the TMR1IF flag is set, and an interrupt will be generated when enabled.

### 10.3 Timer2 Overview

The TMR2 module is an 8-bit timer/counter with an 8-bit period register (PR2). When the TMR2 value rolls over from the period match value to 0h, the TMR2IF flag is set, and an interrupt will be generated when enabled. In counter mode, the clock comes from the RB4/TCLK12 pin, which can also be selected to be the clock for the TMR1 module.

TMR1 can be concatenated to TMR2 to form a 16-bit timer. The TMR2 register is the MSB and TMR1 is the LSB. When in the 16-bit timer mode, there is a corresponding 16-bit period register (PR2:PR1). When the TMR2:TMR1 value rolls over from the period match value to 0h, the TMR1IF flag is set, and an interrupt will be generated when enabled.

### 10.4 Timer3 Overview

The Timer3 module is a 16-bit timer/counter with a 16-bit period register. When the TMR3H:TMR3L value rolls over to 0h, the TMR3IF bit is set and an interrupt will be generated when enabled. In counter mode, the clock comes from the RB5/TCLK3 pin.

When operating in the dual capture mode, the period registers become the second 16-bit capture register.

### 10.5 Role of the Timer/Counters

The timer modules are general purpose, but have dedicated resources associated with them. Timer1 and Timer2 are the time-bases for the two Pulse Width Modulation (PWM) outputs, while Timer3 is the time-base for the two input captures.

# PIC17C4X

---

NOTES:

## 11.0 TIMER0

The Timer0 module consists of a 16-bit timer/counter, TMR0. The high byte is TMR0H and the low byte is TMR0L. A software programmable 8-bit prescaler makes an effective 24-bit overflow timer. The clock source is also software programmable as either the internal instruction clock or the RA1/T0CKI pin. The control bits for this module are in register T0STA (Figure 11-1).

**FIGURE 11-1: T0STA REGISTER (ADDRESS: 05h, UNBANKED)**

R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	U - 0																				
INTEDG	T0SE	T0CS	PS3	PS2	PS1	PS0	—																				
							bit0																				
<p>bit 7: <b>INTEDG:</b> RA0/INT Pin Interrupt Edge Select bit            This bit selects the edge upon which the interrupt is detected            1 = Rising edge of RA0/INT pin generates interrupt            0 = Falling edge of RA0/INT pin generates interrupt</p> <p>bit 6: <b>T0SE:</b> Timer0 Clock Input Edge Select bit            This bit selects the edge upon which TMR0 will increment  <u>When T0CS = 0</u>            1 = Rising edge of RA1/T0CKI pin increments TMR0 and/or generates a T0CKIF interrupt            0 = Falling edge of RA1/T0CKI pin increments TMR0 and/or generates a T0CKIF interrupt  <u>When T0CS = 1</u>            Don't care</p> <p>bit 5: <b>T0CS:</b> Timer0 Clock Source Select bit            This bit selects the clock source for TMR0.            1 = Internal instruction clock cycle (Tcy)            0 = T0CKI pin</p> <p>bit 4-1: <b>PS3:PS0:</b> Timer0 Prescale Selection bits            These bits select the prescale value for TMR0.</p> <table border="1" style="margin-left: 40px;"> <thead> <tr> <th>PS3:PS0</th> <th>Prescale Value</th> </tr> </thead> <tbody> <tr><td>0000</td><td>1:1</td></tr> <tr><td>0001</td><td>1:2</td></tr> <tr><td>0010</td><td>1:4</td></tr> <tr><td>0011</td><td>1:8</td></tr> <tr><td>0100</td><td>1:16</td></tr> <tr><td>0101</td><td>1:32</td></tr> <tr><td>0110</td><td>1:64</td></tr> <tr><td>0111</td><td>1:128</td></tr> <tr><td>1xxx</td><td>1:256</td></tr> </tbody> </table>								PS3:PS0	Prescale Value	0000	1:1	0001	1:2	0010	1:4	0011	1:8	0100	1:16	0101	1:32	0110	1:64	0111	1:128	1xxx	1:256
PS3:PS0	Prescale Value																										
0000	1:1																										
0001	1:2																										
0010	1:4																										
0011	1:8																										
0100	1:16																										
0101	1:32																										
0110	1:64																										
0111	1:128																										
1xxx	1:256																										
<p>bit 0: <b>Unimplemented:</b> Read as '0'</p>																											

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

## 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 (TOIF) is set. The TMR0 interrupt can be masked by clearing the corresponding TMR0 Interrupt Enable bit (TOIE). The TMR0 Interrupt Flag bit (TOIF) 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  $3T_{OSC}$  and  $7T_{OSC}$ . Thus, for example, measuring the interval between two edges (e.g. period) will be accurate within  $\pm 4T_{OSC}$  ( $\pm 121$  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



# PIC17C4X

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

## 12.0 TIMER1, TIMER2, TIMER3, PWMS AND CAPTURES

The PIC17C4X has a wealth of timers and time-based functions to ease the implementation of control applications. These time-base functions include two PWM outputs and two Capture inputs.

Timer1 and Timer2 are two 8-bit incrementing timers, each with a period register (PR1 and PR2 respectively) and separate overflow interrupt flags. Timer1 and Timer2 can operate either as timers (increment on internal Fosc/4 clock) or as counters (increment on falling edge of external clock on pin RB4/TCLK12). They are also software configurable to operate as a single 16-bit timer. These timers are also used as the time-base for the PWM (pulse width modulation) module.

Timer3 is a 16-bit timer/counter consisting of the TMR3H and TMR3L registers. This timer has four other associated registers. Two registers are used as a 16-bit period register or a 16-bit Capture1 register (PR3H/CA1H:PR3L/CA1L). The other two registers are strictly the Capture2 registers (CA2H:CA2L). Timer3 is the time-base for the two 16-bit captures.

TMR3 can be software configured to increment from the internal system clock or from an external signal on the RB5/TCLK3 pin.

Figure 12-1 and Figure 12-2 are the control registers for the operation of Timer1, Timer2, and Timer3, as well as PWM1, PWM2, Capture1, and Capture2.

**FIGURE 12-1: TCON1 REGISTER (ADDRESS: 16h, BANK 3)**

R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0
CA2ED1	CA2ED0	CA1ED1	CA1ED0	T16	TMR3CS	TMR2CS	TMR1CS
bit7				bit0			

R = Readable bit  
W = Writable bit  
-n = Value at POR reset

bit 7-6: **CA2ED1:CA2ED0**: Capture2 Mode Select bits  
00 = Capture on every falling edge  
01 = Capture on every rising edge  
10 = Capture on every 4th rising edge  
11 = Capture on every 16th rising edge

bit 5-4: **CA1ED1:CA1ED0**: Capture1 Mode Select bits  
00 = Capture on every falling edge  
01 = Capture on every rising edge  
10 = Capture on every 4th rising edge  
11 = Capture on every 16th rising edge

bit 3: **T16**: Timer1:Timer2 Mode Select bit  
1 = Timer1 and Timer2 form a 16-bit timer  
0 = Timer1 and Timer2 are two 8-bit timers

bit 2: **TMR3CS**: Timer3 Clock Source Select bit  
1 = TMR3 increments off the falling edge of the RB5/TCLK3 pin  
0 = TMR3 increments off the internal clock

bit 1: **TMR2CS**: Timer2 Clock Source Select bit  
1 = TMR2 increments off the falling edge of the RB4/TCLK12 pin  
0 = TMR2 increments off the internal clock

bit 0: **TMR1CS**: Timer1 Clock Source Select bit  
1 = TMR1 increments off the falling edge of the RB4/TCLK12 pin  
0 = TMR1 increments off the internal clock

# PIC17C4X

**FIGURE 12-2: TCON2 REGISTER (ADDRESS: 17h, BANK 3)**

R - 0	R - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0
CA2OVF	CA1OVF	PWM2ON	PWM1ON	CA1/PR3	TMR3ON	TMR2ON	TMR1ON
							bit0
							bit7

R = Readable bit  
W = Writable bit  
-n = Value at POR reset

bit 7: **CA2OVF**: Capture2 Overflow Status bit  
This bit indicates that the capture value had not been read from the capture register pair (CA2H:CA2L) before the next capture event occurred. The capture register retains the oldest unread capture value (last capture before overflow). Subsequent capture events will not update the capture register with the Timer3 value until the capture register has been read (both bytes).  
1 = Overflow occurred on Capture2 register  
0 = No overflow occurred on Capture2 register

bit 6: **CA1OVF**: Capture1 Overflow Status bit  
This bit indicates that the capture value had not been read from the capture register pair (PR3H/CA2H:PR3L/CA2L) before the next capture event occurred. The capture register retains the oldest unread capture value (last capture before overflow). Subsequent capture events will not update the capture register with the TMR3 value until the capture register has been read (both bytes).  
1 = Overflow occurred on Capture1 register  
0 = No overflow occurred on Capture1 register

bit 5: **PWM2ON**: PWM2 On bit  
1 = PWM2 is enabled (The RB3/PWM2 pin ignores the state of the DDRB<3> bit)  
0 = PWM2 is disabled (The RB3/PWM2 pin uses the state of the DDRB<3> bit for data direction)

bit 4: **PWM1ON**: PWM1 On bit  
1 = PWM1 is enabled (The RB2/PWM1 pin ignores the state of the DDRB<2> bit)  
0 = PWM1 is disabled (The RB2/PWM1 pin uses the state of the DDRB<2> bit for data direction)

bit 3: **CA1/PR3**: CA1/PR3 Register Mode Select bit  
1 = Enables Capture1 (PR3H/CA1H:PR3L/CA1L is the Capture1 register. Timer3 runs without a period register)  
0 = Enables the Period register (PR3H/CA1H:PR3L/CA1L is the Period register for Timer3)

bit 2: **TMR3ON**: Timer3 On bit  
1 = Starts Timer3  
0 = Stops Timer3

bit 1: **TMR2ON**: Timer2 On bit  
This bit controls the incrementing of the Timer2 register. When Timer2:Timer1 form the 16-bit timer (T16 is set), TMR2ON must be set. This allows the MSB of the timer to increment.  
1 = Starts Timer2 (Must be enabled if the T16 bit (TCON1<3>) is set)  
0 = Stops Timer2

bit 0: **TMR1ON**: Timer1 On bit  
When T16 is set (in 16-bit Timer Mode)  
1 = Starts 16-bit Timer2:Timer1  
0 = Stops 16-bit Timer2:Timer1  
  
When T16 is clear (in 8-bit Timer Mode)  
1 = Starts 8-bit Timer1  
0 = Stops 8-bit Timer1



## 12.1 Timer1 and Timer2

### 12.1.1 TIMER1, TIMER2 IN 8-BIT MODE

Both Timer1 and Timer2 will operate in 8-bit mode when the T16 bit is clear. These two timers can be independently configured to increment from the internal instruction cycle clock or from an external clock source on the RB4/TCLK12 pin. The timer clock source is configured by the TMRxCS bit (x = 1 for Timer1 or = 2 for Timer2). When TMRxCS is clear, the clock source is internal and increments once every instruction cycle ( $F_{osc}/4$ ). When TMRxCS is set, the clock source is the RB4/TCLK12 pin, and the timer will increment on every falling edge of the RB4/TCLK12 pin.

The timer increments from 00h until it equals the Period register (PRx). It then resets to 00h at the next increment cycle. The timer interrupt flag is set when the timer is reset. TMR1 and TMR2 have individual interrupt flag bits. The TMR1 interrupt flag bit is latched into TMR1IF, and the TMR2 interrupt flag bit is latched into TMR2IF.

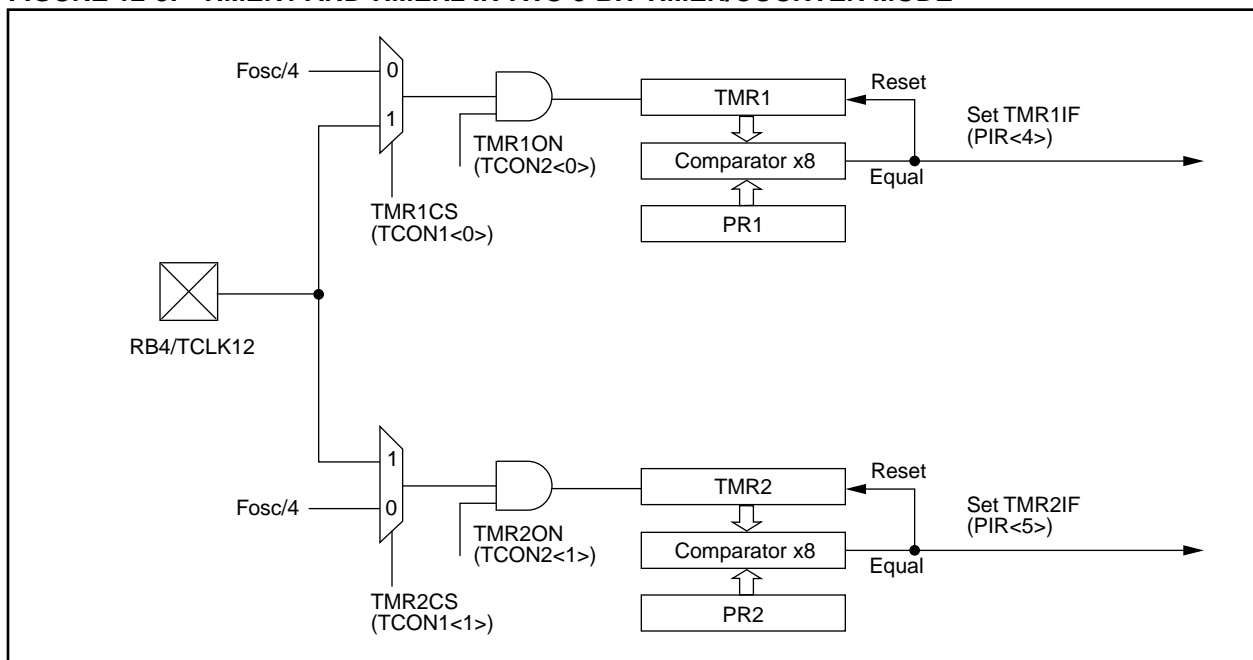
Each timer also has a corresponding interrupt enable bit (TMRxIE). The timer interrupt can be enabled by setting this bit and disabled by clearing this bit. For peripheral interrupts to be enabled, the Peripheral Interrupt Enable bit must be enabled (PEIE is set) and global interrupts must be enabled (GLINTD is cleared).

The timers can be turned on and off under software control. When the Timerx On control bit (TMRxON) is set, the timer increments from the clock source. When TMRxON is cleared, the timer is turned off and cannot cause the timer interrupt flag to be set.

### 12.1.1.1 EXTERNAL CLOCK INPUT FOR TIMER1 OR TIMER2

When TMRxCS is set, the clock source is the RB4/TCLK12 pin, and the timer will increment on every falling edge on the RB4/TCLK12 pin. The TCLK12 input is synchronized with internal phase clocks. This causes a delay from the time a falling edge appears on TCLK12 to the time TMR1 or TMR2 is actually incremented. For the external clock input timing requirements, see the Electrical Specification section.

**FIGURE 12-3: TIMER1 AND TIMER2 IN TWO 8-BIT TIMER/COUNTER MODE**



# PIC17C4X

## 12.1.2 TIMER1 & TIMER2 IN 16-BIT MODE

To select 16-bit mode, the T16 bit must be set. In this mode TMR1 and TMR2 are concatenated to form a 16-bit timer (TMR2:TMR1). The 16-bit timer increments until it matches the 16-bit period register (PR2:PR1). On the following timer clock, the timer value is reset to 0h, and the TMR1IF bit is set.

When selecting the clock source for the 16-bit timer, the TMR1CS bit controls the entire 16-bit timer and TMR2CS is a "don't care." When TMR1CS is clear, the timer increments once every instruction cycle ( $F_{osc}/4$ ). When TMR1CS is set, the timer increments on every falling edge of the RB4/TCLK12 pin. For the 16-bit timer to increment, both TMR1ON and TMR2ON bits must be set (Table 12-1).

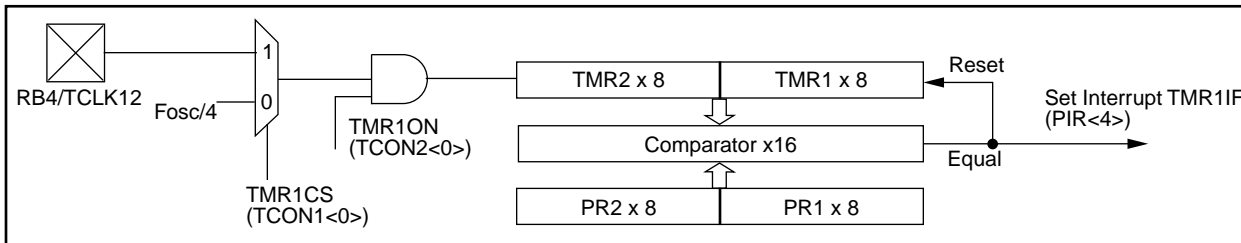
## 12.1.2.1 EXTERNAL CLOCK INPUT FOR TMR1:TMR2

When TMR1CS is set, the 16-bit TMR2:TMR1 increments on the falling edge of clock input TCLK12. The input on the RB4/TCLK12 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 RB4/TCLK12 to the time TMR2:TMR1 is actually incremented. For the external clock input timing requirements, see the Electrical Specification section.

**TABLE 12-1: TURNING ON 16-BIT TIMER**

TMR2ON	TMR1ON	Result
1	1	16-bit timer (TMR2:TMR1) ON
0	1	Only TMR1 increments
x	0	16-bit timer OFF

**FIGURE 12-4: TMR1 AND TMR2 IN 16-BIT TIMER/COUNTER MODE**



**TABLE 12-2: SUMMARY OF TIMER1 AND TIMER2 REGISTERS**

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 3	TCON1	CA2ED1	CA2ED0	CA1ED1	CA1ED0	T16	TMR3CS	TMR2CS	TMR1CS	0000 0000	0000 0000
17h, Bank 3	TCON2	CA2OVF	CA1OVF	PWM2ON	PWM1ON	CA1/PR3	TMR3ON	TMR2ON	TMR1ON	0000 0000	0000 0000
10h, Bank 2	TMR1	Timer1 register								xxxx xxxx	uuuu uuuu
11h, Bank 2	TMR2	Timer2 register								xxxx xxxx	uuuu uuuu
16h, Bank 1	PIR	RBIF	TMR3IF	TMR2IF	TMR1IF	CA2IF	CA1IF	TXIF	RCIF	0000 0010	0000 0010
17h, Bank 1	PIE	RBIE	TMR3IE	TMR2IE	TMR1IE	CA2IE	CA1IE	TXIE	RCIE	0000 0000	0000 0000
07h, Unbanked	INTSTA	PEIF	T0CKIF	T0IF	INTF	PEIE	T0CKIE	T0IE	INTE	0000 0000	0000 0000
06h, Unbanked	CPUSTA	—	—	STKAV	GLINTD	$\overline{TO}$	$\overline{PD}$	—	—	--11 11--	--11 qq--
14h, Bank 2	PR1	Timer1 period register								xxxx xxxx	uuuu uuuu
15h, Bank 2	PR2	Timer2 period register								xxxx xxxx	uuuu uuuu
10h, Bank 3	PW1DCL	DC1	DC0	—	—	—	—	—	—	xx-- ----	uu-- ----
11h, Bank 3	PW2DCL	DC1	DC0	TM2PW2	—	—	—	—	—	xx0- ----	uu0- ----
12h, Bank 3	PW1DCH	DC9	DC8	DC7	DC6	DC5	DC4	DC3	DC2	xxxx xxxx	uuuu uuuu
13h, Bank 3	PW2DCH	DC9	DC8	DC7	DC6	DC5	DC4	DC3	DC2	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 Timer1 or Timer2.

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

## 12.1.3 USING PULSE WIDTH MODULATION (PWM) OUTPUTS WITH TMR1 AND TMR2

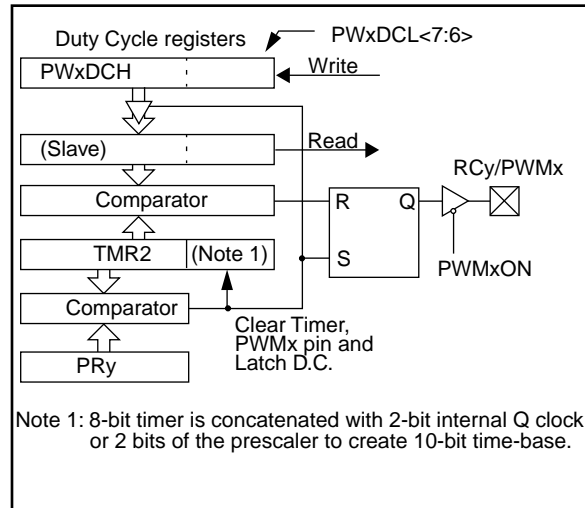
Two high speed pulse width modulation (PWM) outputs are provided. The PWM1 output uses Timer1 as its time-base, while PWM2 may be software configured to use either Timer1 or Timer2 as the time-base. The PWM outputs are on the RB2/PWM1 and RB3/PWM2 pins.

Each PWM output has a maximum resolution of 10-bits. At 10-bit resolution, the PWM output frequency is 24.4 kHz (@ 25 MHz clock) and at 8-bit resolution the PWM output frequency is 97.7 kHz. The duty cycle of the output can vary from 0% to 100%.

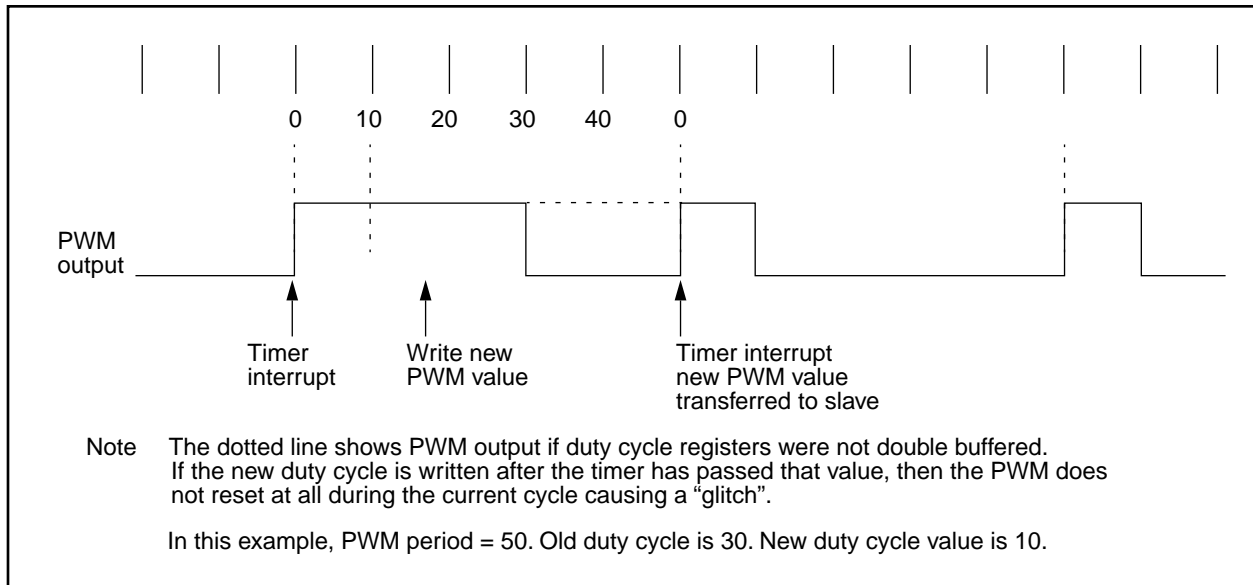
Figure 12-5 shows a simplified block diagram of the PWM module. The duty cycle register is double buffered for glitch free operation. Figure 12-6 shows how a glitch could occur if the duty cycle registers were not double buffered.

The user needs to set the PWM1ON bit (TCON2<4>) to enable the PWM1 output. When the PWM1ON bit is set, the RB2/PWM1 pin is configured as PWM1 output and forced as an output irrespective of the data direction bit (DDRB<2>). When the PWM1ON bit is clear, the pin behaves as a port pin and its direction is controlled by its data direction bit (DDRB<2>). Similarly, the PWM2ON (TCON2<5>) bit controls the configuration of the RB3/PWM2 pin.

**FIGURE 12-5: SIMPLIFIED PWM BLOCK DIAGRAM**



**FIGURE 12-6: PWM OUTPUT**



# PIC17C4X

## 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:

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

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 ±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).

## 12.1.3.3.1 MAX RESOLUTION/FREQUENCY FOR EXTERNAL CLOCK INPUT

The use of an external clock for the PWM time-base (Timer1 or Timer2) limits the PWM output to a maximum resolution of 8-bits. The PWxDCL<7:6> bits must be kept cleared. Use of any other value will distort the PWM output. All resolutions are supported when internal clock mode is selected. The maximum attainable frequency is also lower. This is a result of the timing requirements of an external clock input for a timer (see the Electrical Specification section). The maximum PWM frequency, when the timers clock source is the RB4/TCLK12 pin, is shown in Table 12-3 (standard resolution mode).

## 12.2 Timer3

Timer3 is a 16-bit timer consisting of the TMR3H and TMR3L registers. TMR3H is the high byte of the timer and TMR3L is the low byte. This timer has an associated 16-bit period register (PR3H/CA1H:PR3L/CA1L). This period register can be software configured to be a second 16-bit capture register.

When the TMR3CS bit (TCON1<2>) is clear, the timer increments every instruction cycle ( $F_{osc}/4$ ). When TMR3CS is set, the timer increments on every falling edge of the RB5/TCLK3 pin. In either mode, the TMR3ON bit must be set for the timer to increment. When TMR3ON is clear, the timer will not increment or set the TMR3IF bit.

Timer3 has two modes of operation, depending on the CA1/PR3 bit (TCON2<3>). These modes are:

- One capture and one period register mode
- Dual capture register mode

The PIC17C4X has up to two 16-bit capture registers that capture the 16-bit value of TMR3 when events are detected on capture pins. There are two capture pins (RB0/CAP1 and RB1/CAP2), one for each capture register. The capture pins are multiplexed with PORTB pins. An event can be:

- a rising edge
- a falling edge
- every 4th rising edge
- every 16th rising edge

Each 16-bit capture register has an interrupt flag associated with it. The flag is set when a capture is made. The capture module is truly part of the Timer3 block. Figure 12-7 and Figure 12-8 show the block diagrams for the two modes of operation.

**TABLE 12-4: REGISTERS/BITS ASSOCIATED WITH PWM**

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 3	TCON1	CA2ED1	CA2ED0	CA1ED1	CA1ED0	T16	TMR3CS	TMR2CS	TMR1CS	0000 0000	0000 0000
17h, Bank 3	TCON2	CA2OVF	CA1OVF	PWM2ON	PWM1ON	CA1/PR3	TMR3ON	TMR2ON	TMR1ON	0000 0000	0000 0000
10h, Bank 2	TMR1	Timer1 register								xxxx xxxx	uuuu uuuu
11h, Bank 2	TMR2	Timer2 register								xxxx xxxx	uuuu uuuu
16h, Bank 1	PIR	RBIF	TMR3IF	TMR2IF	TMR1IF	CA2IF	CA1IF	TXIF	RCIF	0000 0010	0000 0010
17h, Bank 1	PIE	RBIE	TMR3IE	TMR2IE	TMR1IE	CA2IE	CA1IE	TXIE	RCIE	0000 0000	0000 0000
07h, Unbanked	INTSTA	PEIF	T0CKIF	T0IF	INTF	PEIE	T0CKIE	T0IE	INTE	0000 0000	0000 0000
06h, Unbanked	CPUSTA	—	—	STKAV	GLINTD	T0	PD	—	—	--11 11--	--11 qg--
10h, Bank 3	PW1DCL	DC1	DC0	—	—	—	—	—	—	xx-- ----	uu-- ----
11h, Bank 3	PW2DCL	DC1	DC0	TM2PW2	—	—	—	—	—	xx0- ----	uu0- ----
12h, Bank 3	PW1DCH	DC9	DC8	DC7	DC6	DC5	DC4	DC3	DC2	xxxx xxxx	uuuu uuuu
13h, Bank 3	PW2DCH	DC9	DC8	DC7	DC6	DC5	DC4	DC3	DC2	xxxx xxxx	uuuu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented read as '0', q = value depends on conditions, shaded cells are not used by PWM.

## 12.2.1 ONE CAPTURE AND ONE PERIOD REGISTER MODE

In this mode registers PR3H/CA1H and PR3L/CA1L constitute a 16-bit period register. A block diagram is shown in Figure 12-7. The timer increments until it equals the period register and then resets to 0000h. TMR3 Interrupt Flag bit (TMR3IF) is set at this point. This interrupt can be disabled by clearing the TMR3 Interrupt Enable bit (TMR3IE). TMR3IF must be cleared in software.

This mode is selected if control bit CA1/PR3 is clear. In this mode, the Capture1 register, consisting of high byte (PR3H/CA1H) and low byte (PR3L/CA1L), is configured as the period control register for TMR3. Capture1 is disabled in this mode, and the corresponding Interrupt bit CA1IF is never set. TMR3 increments until it equals the value in the period register and then resets to 0000h.

Capture2 is active in this mode. The CA2ED1 and CA2ED0 bits determine the event on which capture will occur. The possible events are:

- Capture on every falling edge
- Capture on every rising edge
- Capture every 4th rising edge
- Capture every 16th rising edge

When a capture takes place, an interrupt flag is latched into the CA2IF bit. This interrupt can be enabled by setting the corresponding mask bit CA2IE. The Peripheral Interrupt Enable bit (PEIE) must be set and the Global Interrupt Disable bit (GLINTD) must be cleared for the interrupt to be acknowledged. The CA2IF interrupt flag bit must be cleared in software.

When the capture prescale select is changed, the prescaler is not reset and an event may be generated. Therefore, the first capture after such a change will be ambiguous. However, it sets the time-base for the next capture. The prescaler is reset upon chip reset.

Capture pin RB1/CAP2 is a multiplexed pin. When used as a port pin, Capture2 is not disabled. However, the user can simply disable the Capture2 interrupt by clearing CA2IE. If RB1/CAP2 is used as an output pin, the user can activate a capture by writing to the port pin. This may be useful during development phase to emulate a capture interrupt.

The input on capture pin RB1/CAP2 is synchronized internally to internal phase clocks. This imposes certain restrictions on the input waveform (see the Electrical Specification section for timing).

The Capture2 overflow status flag bit is double buffered. The master bit is set if one captured word is already residing in the Capture2 register and another "event" has occurred on the RB1/CA2 pin. The new event will not transfer the Timer3 value to the capture register, protecting the previous unread capture value. When the user reads both the high and the low bytes (in any order) of the Capture2 register, the master overflow bit is transferred to the slave overflow bit (CA2OVF) and then the master bit is reset. The user can then read TCON2 to determine the value of CA2OVF.

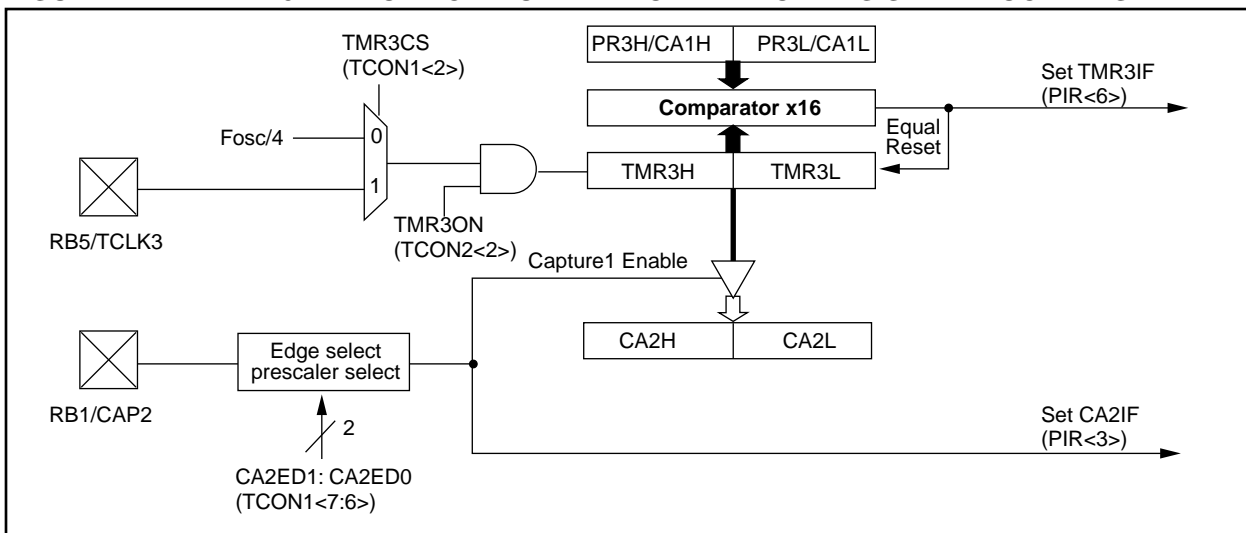
The recommended sequence to read capture registers and capture overflow flag bits is shown in Example 12-1.

### EXAMPLE 12-1: SEQUENCE TO READ CAPTURE REGISTERS

```

MOVLB 3           ;Select Bank 3
MOVPF CA2L,LO_BYTE ;Read Capture2 low
                  ;byte, store in LO_BYTE
MOVPF CA2H,HI_BYTE ;Read Capture2 high
                  ;byte, store in HI_BYTE
MOVPF TCON2,STAT_VAL ;Read TCON2 into file
                  ;STAT_VAL
    
```

**FIGURE 12-7: TIMER3 WITH ONE CAPTURE AND ONE PERIOD REGISTER BLOCK DIAGRAM**



## 12.2.2 DUAL CAPTURE REGISTER MODE

This mode is selected by setting CA1/PR3. A block diagram is shown in Figure 12-8. In this mode, TMR3 runs without a period register and increments from 0000h to FFFFh and rolls over to 0000h. The TMR3 interrupt Flag (TMR3IF) is set on this roll over. The TMR3IF bit must be cleared in software.

Registers PR3H/CA1H and PR3L/CA1L make a 16-bit capture register (Capture1). It captures events on pin RB0/CAP1. Capture mode is configured by the CA1ED1 and CA1ED0 bits. Capture1 Interrupt Flag bit (CA1IF) is set on the capture event. The corresponding interrupt mask bit is CA1IE. The Capture1 Overflow Status bit is CA1OVF.

The Capture2 overflow status flag bit is double buffered. The master bit is set if one captured word is already residing in the Capture2 register and another "event" has occurred on the RB1/CA2 pin. The new event will not transfer the TMR3 value to the capture register which protects the previous unread capture value. When the user reads both the high and the low bytes (in any order) of the Capture2 register, the master overflow bit is transferred to the slave overflow bit (CA2OVF) and then the master bit is reset. The user can then read TCON2 to determine the value of CA2OVF.

The operation of the Capture1 feature is identical to Capture2 (as described in Section 12.2.1).

**FIGURE 12-8: TIMER3 WITH TWO CAPTURE REGISTERS BLOCK DIAGRAM**



**TABLE 12-5: REGISTERS ASSOCIATED WITH CAPTURE**

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 3	TCON1	CA2ED1	CA2ED0	CA1ED1	CA1ED0	T16	TMR3CS	TMR2CS	TMR1CS	0000 0000	0000 0000
17h, Bank 3	TCON2	CA2OVF	CA1OVF	PWM2ON	PWM1ON	CA1/PR3	TMR3ON	TMR2ON	TMR1ON	0000 0000	0000 0000
12h, Bank 2	TMR3L	TMR3 register; low byte								xxxx xxxx	uuuu uuuu
13h, Bank 2	TMR3H	TMR3 register; high byte								xxxx xxxx	uuuu uuuu
16h, Bank 1	PIR	RBIF	TMR3IF	TMR2IF	TMR1IF	CA2IF	CA1IF	TXIF	RCIF	0000 0010	0000 0010
17h, Bank 1	PIE	RBIE	TMR3IE	TMR2IE	TMR1IE	CA2IE	CA1IE	TXIE	RCIE	0000 0000	0000 0000
07h, Unbanked	INTSTA	PEIF	T0CKIF	T0IF	INTF	PEIE	T0CKIE	T0IE	INTE	0000 0000	0000 0000
06h, Unbanked	CPUSTA	—	—	STKAV	GLINTD	T0	PD	—	—	--11 11--	--11 qq--
16h, Bank 2	PR3L/CA1L	Timer3 period register, low byte/capture1 register, low byte								xxxx xxxx	uuuu uuuu
17h, Bank 2	PR3H/CA1H	Timer3 period register, high byte/capture1 register, high byte								xxxx xxxx	uuuu uuuu
14h, Bank 3	CA2L	Capture2 low byte								xxxx xxxx	uuuu uuuu
15h, Bank 3	CA2H	Capture2 high byte								xxxx xxxx	uuuu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented read as '0', q - value depends on condition, shaded cells are not used by Capture.

Note 1: Other (non power-up) resets include: external reset through MCLR and WDT Timer Reset.

# PIC17C4X

## 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.

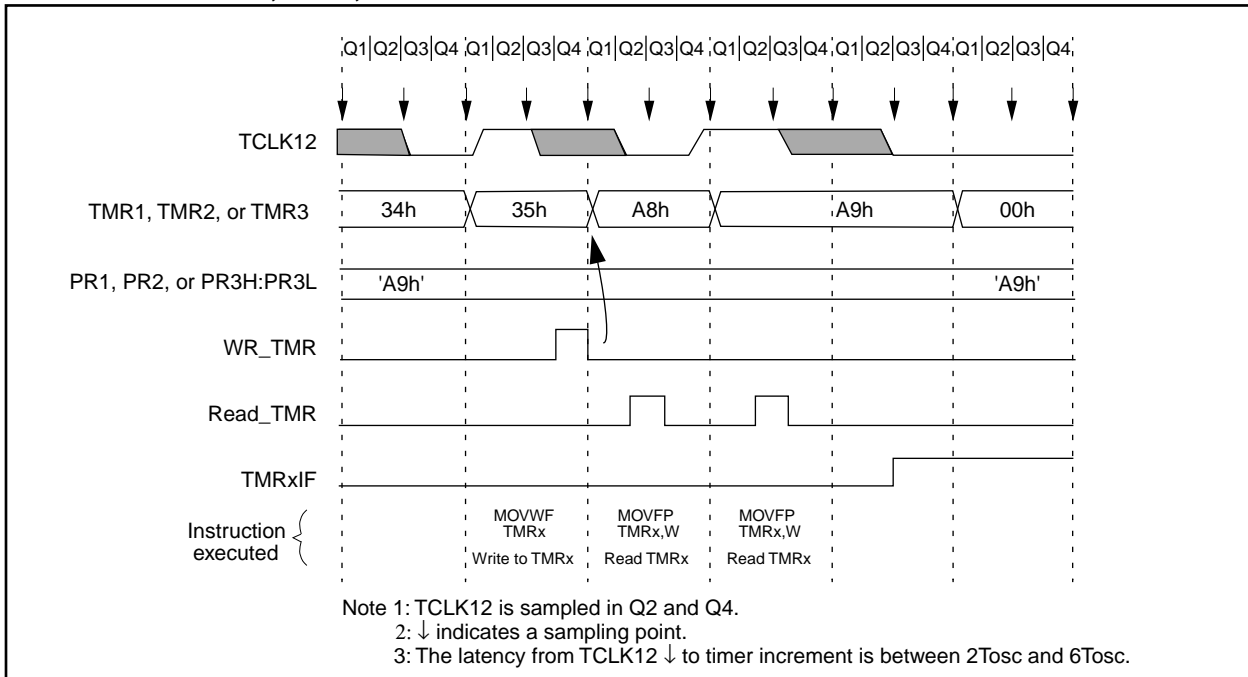
## EXAMPLE 12-2: WRITING TO TMR3

```
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

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

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





**FIGURE 12-10: TMR1, TMR2, AND TMR3 OPERATION IN TIMER MODE**



**TABLE 12-6: SUMMARY OF TMR1, TMR2, AND TMR3 REGISTERS**

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 3	TCON1	CA2ED1	CA2ED0	CA1ED1	CA1ED0	T16	TMR3CS	TMR2CS	TMR1CS	0000 0000	0000 0000
17h, Bank 3	TCON2	CA2OVF	CA1OVF	PWM2ON	PWM1ON	CA1/PR3	TMR3ON	TMR2ON	TMR1ON	0000 0000	0000 0000
10h, Bank 2	TMR1	Timer1 register								xxxx xxxx	uuuu uuuu
11h, Bank 2	TMR2	Timer2 register								xxxx xxxx	uuuu uuuu
12h, Bank 2	TMR3L	TMR3 register; low byte								xxxx xxxx	uuuu uuuu
13h, Bank 2	TMR3H	TMR3 register; high byte								xxxx xxxx	uuuu uuuu
16h, Bank 1	PIR	RBIF	TMR3IF	TMR2IF	TMR1IF	CA2IF	CA1IF	TXIF	RCIF	0000 0010	0000 0010
17h, Bank 1	PIE	RBIE	TMR3IE	TMR2IE	TMR1IE	CA2IE	CA1IE	TXIE	RCIE	0000 0000	0000 0000
07h, Unbanked	INTSTA	PEIF	T0CKIF	T0IF	INTF	PEIE	T0CKIE	T0IE	INTE	0000 0000	0000 0000
06h, Unbanked	CPUSTA	—	—	STKAV	GLINTD	T0	PD	—	—	--11 11--	--11 qq--
14h, Bank 2	PR1	Timer1 period register								xxxx xxxx	uuuu uuuu
15h, Bank 2	PR2	Timer2 period register								xxxx xxxx	uuuu uuuu
16h, Bank 2	PR3L/CA1L	Timer3 period/capture1 register; low byte								xxxx xxxx	uuuu uuuu
17h, Bank 2	PR3H/CA1H	Timer3 period/capture1 register; high byte								xxxx xxxx	uuuu uuuu
10h, Bank 3	PW1DCL	DC1	DC0	—	—	—	—	—	—	xx-- ----	uu-- ----
11h, Bank 3	PW2DCL	DC1	DC0	TM2PW2	—	—	—	—	—	xx0- ----	uu0- ----
12h, Bank 3	PW1DCH	DC9	DC8	DC7	DC6	DC5	DC4	DC3	DC2	xxxx xxxx	uuuu uuuu
13h, Bank 3	PW2DCH	DC9	DC8	DC7	DC6	DC5	DC4	DC3	DC2	xxxx xxxx	uuuu uuuu
14h, Bank 3	CA2L	Capture2 low byte								xxxx xxxx	uuuu uuuu
15h, Bank 3	CA2H	Capture2 high byte								xxxx xxxx	uuuu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented read as '0', q - value depends on condition, shaded cells are not used by TMR1, TMR2 or TMR3.

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

# PIC17C4X

---

NOTES:

## 13.0 UNIVERSAL SYNCHRONOUS ASYNCHRONOUS RECEIVER TRANSMITTER (USART) MODULE

The USART module is a serial I/O module. The USART can be configured as a full duplex asynchronous system that can communicate with peripheral devices such as CRT terminals and personal computers, or it can be configured as a half duplex synchronous system that can communicate with peripheral devices such as A/D or D/A integrated circuits, Serial EEPROMs etc. The USART can be configured in the following modes:

- Asynchronous (full duplex)
- Synchronous - Master (half duplex)
- Synchronous - Slave (half duplex)

The SPEN (RCSTA<7>) bit has to be set in order to configure RA4 and RA5 as the Serial Communication Interface.

The USART module will control the direction of the RA4/RX/DT and RA5/TX/CK pins, depending on the states of the USART configuration bits in the RCSTA and TXSTA registers. The bits that control I/O direction are:

- SPEN
- TXEN
- SREN
- CREN
- CSRC

The Transmit Status And Control Register is shown in Figure 13-1, while the Receive Status And Control Register is shown in Figure 13-2.

**FIGURE 13-1: TXSTA REGISTER (ADDRESS: 15h, BANK 0)**



# PIC17C4X

**FIGURE 13-2: RCSTA REGISTER (ADDRESS: 13h, BANK 0)**

	R/W - 0	R/W - 0	R/W - 0	R/W - 0	U - 0	R - 0	R - 0	R - x
bit7	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D
								bit 0

R = Readable bit  
 W = Writable bit  
 -n = Value at POR reset  
 (x = unknown)

bit 7: **SPEN:** Serial Port Enable bit  
 1 = Configures RA5/RX/DT and RA4/TX/CK pins as serial port pins  
 0 = Serial port disabled

bit 6: **RX9:** 9-bit Receive Enable bit  
 1 = Selects 9-bit reception  
 0 = Selects 8-bit reception

bit 5: **SREN:** Single Receive Enable bit  
 This bit enables the reception of a single byte. After receiving the byte, this bit is automatically cleared.  
Synchronous mode:  
 1 = Enable reception  
 0 = Disable reception  
 Note: This bit is ignored in synchronous slave reception.  
Asynchronous mode:  
 Don't care

bit 4: **CREN:** Continuous Receive Enable bit  
 This bit enables the continuous reception of serial data.  
Asynchronous mode:  
 1 = Enable reception  
 0 = Disables reception  
Synchronous mode:  
 1 = Enables continuous reception until CREN is cleared (CREN overrides SREN)  
 0 = Disables continuous reception

bit 3: **Unimplemented:** Read as '0'

bit 2: **FERR:** Framing Error bit  
 1 = Framing error (Updated by reading RCREG)  
 0 = No framing error

bit 1: **OERR:** Overrun Error bit  
 1 = Overrun (Cleared by clearing CREN)  
 0 = No overrun error

bit 0: **RX9D:** 9th bit of receive data (can be the software calculated parity bit)

**FIGURE 13-3: USART TRANSMIT**



**FIGURE 13-4: USART RECEIVE**



# PIC17C4X

## 13.1 USART Baud Rate Generator (BRG)

The BRG supports both the Asynchronous and Synchronous modes of the USART. It is a dedicated 8-bit baud rate generator. The SPBRG register controls the period of a free running 8-bit timer. Table 13-1 shows the formula for computation of the baud rate for different USART modes. These only apply when the USART is in synchronous master mode (internal clock) and asynchronous mode.

Given the desired baud rate and Fosc, the nearest integer value between 0 and 255 can be calculated using the formula below. The error in baud rate can then be determined.

**TABLE 13-1: BAUD RATE FORMULA**

SYNC	Mode	Baud Rate
0	Asynchronous	$F_{OSC}/(64(X+1))$
1	Synchronous	$F_{OSC}/(4(X+1))$

X = value in SPBRG (0 to 255)

Example 13-1 shows the calculation of the baud rate error for the following conditions:

Fosc = 16 MHz  
 Desired Baud Rate = 9600  
 SYNC = 0

### EXAMPLE 13-1: CALCULATING BAUD RATE ERROR

$$\begin{aligned} \text{Desired Baud rate} &= F_{osc} / (64 (X + 1)) \\ 9600 &= 16000000 / (64 (X + 1)) \\ X &= 25.042 = 25 \\ \text{Calculated Baud Rate} &= 16000000 / (64 (25 + 1)) \\ &= 9615 \\ \text{Error} &= \frac{(\text{Calculated Baud Rate} - \text{Desired Baud Rate})}{\text{Desired Baud Rate}} \\ &= (9615 - 9600) / 9600 \\ &= 0.16\% \end{aligned}$$

Writing a new value to the SPBRG, causes the BRG timer to be reset (or cleared), this ensures that the BRG does not wait for a timer overflow before outputting the new baud rate.

**TABLE 13-2: REGISTERS ASSOCIATED WITH BAUD RATE GENERATOR**

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)
13h, Bank 0	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00u
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 by the Baud Rate Generator.

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

**TABLE 13-3: BAUD RATES FOR SYNCHRONOUS 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	—	—	NA	—	—	NA	—	—
2.4	NA	—	—	NA	—	—	NA	—	—	NA	—	—
9.6	NA	—	—	NA	—	—	NA	—	—	NA	—	—
19.2	NA	—	—	NA	—	—	19.53	+1.73	255	19.23	+0.16	207
76.8	77.10	+0.39	106	77.16	+0.47	80	76.92	+0.16	64	76.92	+0.16	51
96	95.93	-0.07	85	96.15	+0.16	64	96.15	+0.16	51	95.24	-0.79	41
300	294.64	-1.79	27	297.62	-0.79	20	294.1	-1.96	16	307.69	+2.56	12
500	485.29	-2.94	16	480.77	-3.85	12	500	0	9	500	0	7
HIGH	8250	—	0	6250	—	0	5000	—	0	4000	—	0
LOW	32.22	—	255	24.41	—	255	19.53	—	255	15.625	—	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	—	—	NA	—	—
1.2	NA	—	—	NA	—	—	NA	—	—
2.4	NA	—	—	NA	—	—	NA	—	—
9.6	9.766	+1.73	255	9.622	+0.23	185	9.6	0	131
19.2	19.23	+0.16	129	19.24	+0.23	92	19.2	0	65
76.8	75.76	-1.36	32	77.82	+1.32	22	79.2	+3.13	15
96	96.15	+0.16	25	94.20	-1.88	18	97.48	+1.54	12
300	312.5	+4.17	7	298.3	-0.57	5	316.8	+5.60	3
500	500	0	4	NA	—	—	NA	—	—
HIGH	2500	—	0	1789.8	—	0	1267	—	0
LOW	9.766	—	255	6.991	—	255	4.950	—	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	NA	—	—	NA	—	—	0.303	+1.14	26
1.2	NA	—	—	1.202	+0.16	207	1.170	-2.48	6
2.4	NA	—	—	2.404	+0.16	103	NA	—	—
9.6	9.622	+0.23	92	9.615	+0.16	25	NA	—	—
19.2	19.04	-0.83	46	19.24	+0.16	12	NA	—	—
76.8	74.57	-2.90	11	83.34	+8.51	2	NA	—	—
96	99.43	-3.57	8	NA	—	—	NA	—	—
300	298.3	-0.57	2	NA	—	—	NA	—	—
500	NA	—	—	NA	—	—	NA	—	—
HIGH	894.9	—	0	250	—	0	8.192	—	0
LOW	3.496	—	255	0.976	—	255	0.032	—	255

# PIC17C4X

**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



## 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<sub>CY</sub> 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 than 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.

# PIC17C4X

**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  $\overline{MCLR}$  and Watchdog Timer Reset.

## 13.2.2 USART ASYNCHRONOUS RECEIVER

The receiver block diagram is shown in Figure 13-4. The data comes in the RA4/RX/DT pin and drives the data recovery block. The data recovery block is actually a high speed shifter operating at 16 times the baud rate, whereas the main receive serial shifter operates at the bit rate or at FOSC.

Once asynchronous mode is selected, reception is enabled by setting bit CREN (RCSTA<4>).

The heart of the receiver is the receive (serial) shift register (RSR). After sampling the stop bit, the received data in the RSR is transferred to the 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 cleared by the hardware. It is cleared 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 begin shifting to the RSR. On detection of the stop bit of the third byte, if the RCREG is still full, then the overrun error bit, OERR (RCSTA<1>) will be 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 which is done by resetting the receive logic (CREN is set). If the OERR bit is set, transfers from the RSR to RCREG are inhibited, so it is essential to clear the OERR bit if it is set. The framing error bit FERR (RCSTA<2>) is set if a stop bit is not detected.

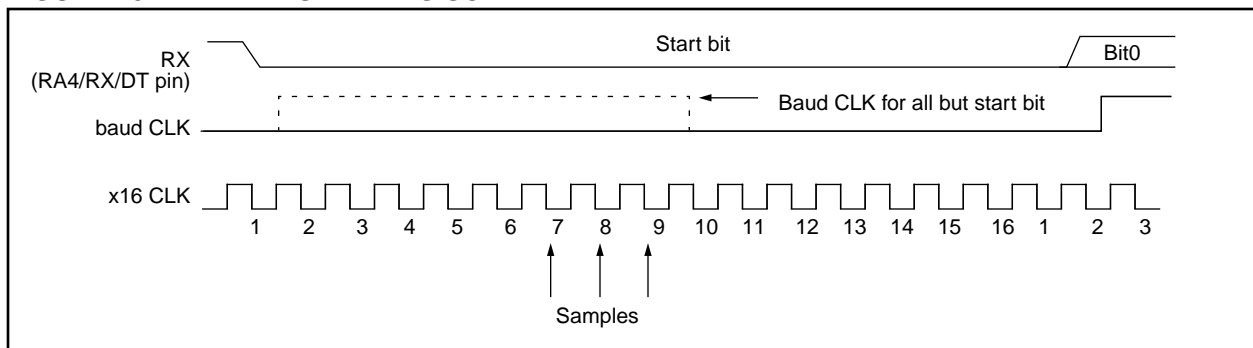
**Note:** The FERR and the 9th receive bit are buffered the same way as the receive data. Reading the RCREG register will allow the RX9D and FERR bits to be loaded with values for the next received Received data; therefore, it is essential for the user to read the RCSTA register before reading RCREG in order not to lose the old FERR and RX9D information.

## 13.2.3 SAMPLING

The data on the RA4/RX/DT pin is sampled three times by a majority detect circuit to determine if a high or a low level is present at the RA4/RX/DT pin. The sampling is done on the seventh, eighth and ninth falling edges of a x16 clock (Figure 11-3).

The x16 clock is a free running clock, and the three sample points occur at a frequency of every 16 falling edges.

**FIGURE 13-7: RX PIN SAMPLING SCHEME**



# PIC17C4X

Steps to follow when setting up an Asynchronous Reception:

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 RCIE bit.
4. If 9-bit reception is desired, then set the RX9 bit.
5. Enable the reception by setting the CREN bit.
6. The RCIF bit will be set when reception completes and an interrupt will be generated if the RCIE bit was set.

7. Read RCSTA to get the ninth bit (if enabled) and FERR bit to determine if any error occurred during reception.
8. Read RCREG for the 8-bit received data.
9. If an overrun error occurred, clear the error by clearing the OERR bit.

**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 13-8: ASYNCHRONOUS RECEPTION**



**TABLE 13-6: REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION**

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
14h, Bank 0	RCREG	RX7	RX6	RX5	RX4	RX3	RX2	RX1	RX0	xxxx xxxxx	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 xxxxx	uuuu uuuu

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

Note 1: Other (non power-up) resets include: external reset through MCLR and Watchdog Timer Reset.

## 13.3 USART Synchronous Master Mode

In Master Synchronous mode, the data is transmitted in a half-duplex manner; i.e. transmission and reception do not occur at the same time: when transmitting data, the reception is inhibited and vice versa. The synchronous mode is entered by setting the SYNC (TXSTA<4>) bit. In addition, the SPEN (RCSTA<7>) bit is set in order to configure the RA5 and RA4 I/O ports to CK (clock) and DT (data) lines respectively. The Master mode indicates that the processor transmits the master clock on the CK line. The Master mode is entered by setting the CSRC (TXSTA<7>) bit.

### 13.3.1 USART SYNCHRONOUS MASTER TRANSMISSION

The USART transmitter block diagram is shown in Figure 13-3. The heart of the transmitter is the transmit (serial) shift register (TSR). The shift register obtains its data from the read/write transmit buffer TXREG. TXREG is loaded with data in software. The TSR is not loaded until the last bit has been transmitted from the previous load. As soon as the last bit is transmitted, the TSR is loaded with new data from TXREG (if available). Once TXREG transfers the data to the TSR (occurs in one T<sub>CY</sub> at the end of the current BRG cycle), TXREG is empty and the TXIF (PIR<1>) bit is set. This interrupt can be enabled/disabled by setting/clearing the TXIE bit (PIE<1>). TXIF will be set regardless of the state of bit TXIE and cannot be cleared in software. It will reset only when new data is loaded into TXREG. While TXIF indicates the status of TXREG, TRMT (TXSTA<1>) shows the status of the TSR. TRMT is a read only bit which is set when the TSR is empty. No interrupt logic is tied to this bit, so the user has to poll this bit in order to determine if the TSR is empty. The TSR is not mapped in data memory, so it is not available to the user.

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

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

RA4/RX/DT pin reverts to a hi-impedance state (for a reception). The RA5/TX/CK pin will remain an output if the CSRC bit is set (internal clock). The transmitter logic is not reset, although it is disconnected from the pins. In order to reset the transmitter, the user has to clear the TXEN bit. If the SREN bit is set (to interrupt an ongoing transmission and receive a single word), then after the single word is received, SREN will be cleared and the serial port will revert back to transmitting, since the TXEN bit is still set. The DT line will immediately switch from hi-impedance receive mode to transmit and start driving. To avoid this, TXEN should be cleared.

In order to select 9-bit transmission, the TX9 (TXSTA<6>) bit should be set and the ninth bit should be written to TX9D (TXSTA<0>). The ninth bit must be written before writing the 8-bit data to TXREG. This is because a data write to TXREG can result in an immediate transfer of the data to the TSR (if the TSR is empty). If the TSR was empty and TXREG was written before writing the "new" TX9D, the "present" value of TX9D is loaded.

Steps to follow when setting up a Synchronous Master Transmission:

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

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

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

# PIC17C4X

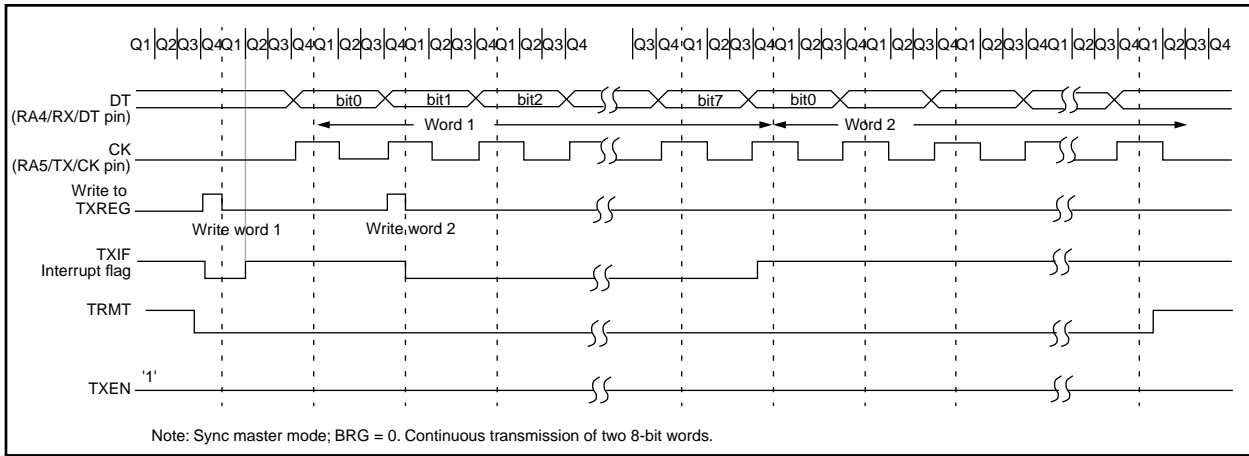
**TABLE 13-7: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER TRANSMISSION**

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	TX7	TX6	TX5	TX4	TX3	TX2	TX1	TX0	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

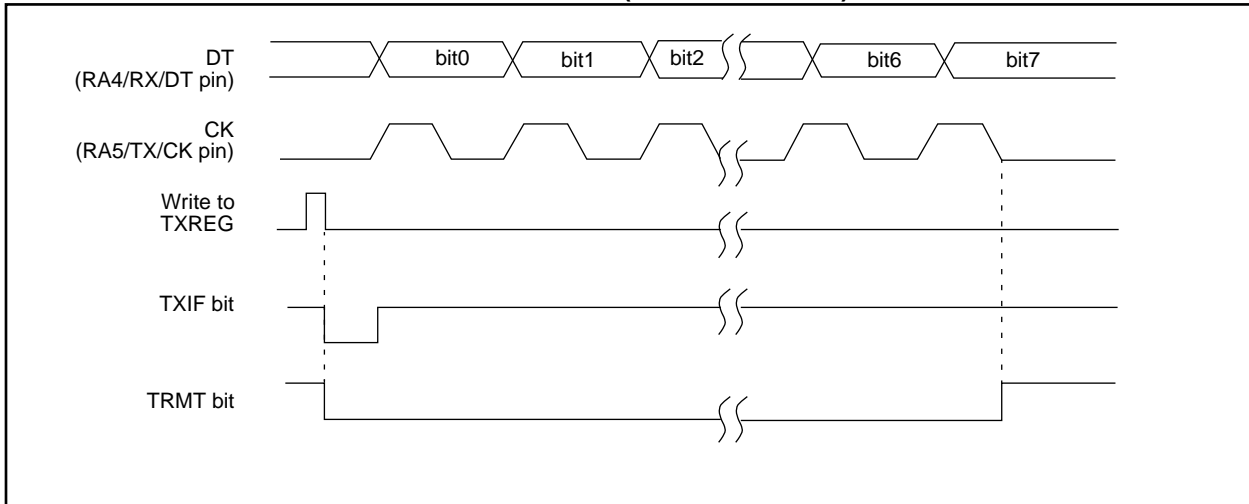
Legend: x = unknown, u = unchanged, - = unimplemented read as a '0', shaded cells are not used for synchronous master transmission.

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

**FIGURE 13-9: SYNCHRONOUS TRANSMISSION**



**FIGURE 13-10: SYNCHRONOUS TRANSMISSION (THROUGH TXEN)**



## 13.3.2 USART SYNCHRONOUS MASTER RECEPTION

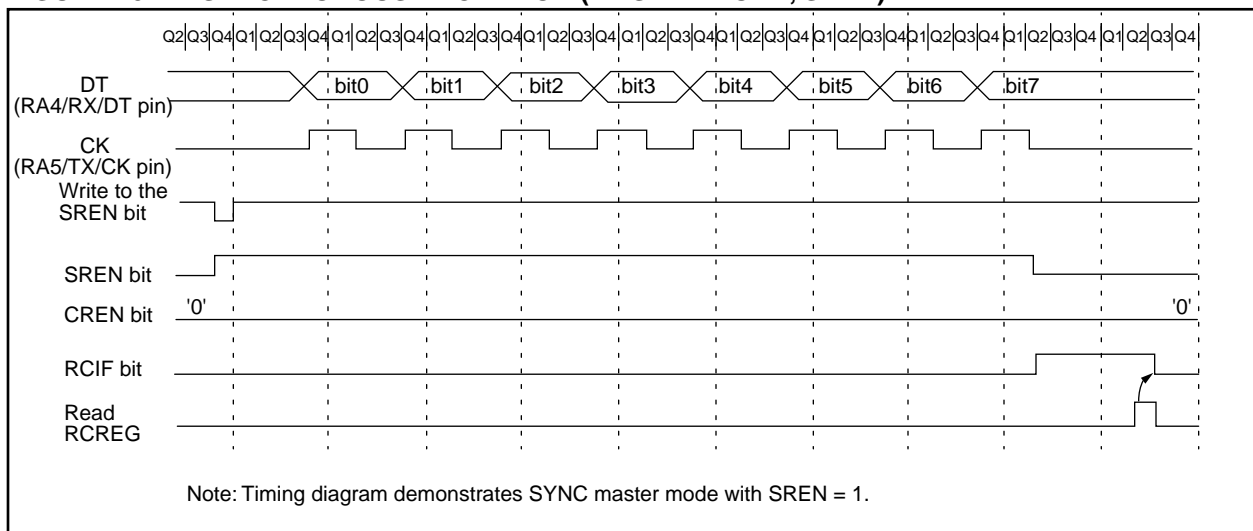
Once synchronous mode is selected, reception is enabled by setting either the SREN (RCSTA<5>) bit or the CREN (RCSTA<4>) bit. Data is sampled on the RA4/RX/DT pin on the falling edge of the clock. If SREN is set, then only a single word is received. If CREN is set, the reception is continuous until CREN 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)**



# PIC17C4X

**TABLE 13-8: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER RECEPTION**

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
14h, Bank 0	RCREG	RX7	RX6	RX5	RX4	RX3	RX2	RX1	RX0	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 synchronous master reception.

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



## 13.4 USART Synchronous Slave Mode

The synchronous slave mode differs from the master mode in the fact that the shift clock is supplied externally at the RA5/TX/CK pin (instead of being supplied internally in the master mode). This allows the device to transfer or receive data in the SLEEP mode. The slave mode is entered by clearing the CSRC (TXSTA<7>) bit.

### 13.4.1 USART SYNCHRONOUS SLAVE TRANSMIT

The operation of the sync master and slave modes are identical except in the case of the SLEEP mode.

If two words are written to TXREG and then the SLEEP instruction executes, the following will occur. The first word will immediately transfer to the TSR and will transmit as the shift clock is supplied. The second word will remain in TXREG. TXIF will not be set. When the first word has been shifted out of TSR, TXREG will transfer the second word to the TSR and the TXIF flag will now be set. If TXIE is enabled, the interrupt will wake the chip from SLEEP and if the global interrupt is enabled, then the program will branch to interrupt vector (0020h).

Steps to follow when setting up a Synchronous Slave Transmission:

1. Enable the synchronous slave serial port by setting the SYNC and SPEN bits and clearing the CSRC bit.
2. Clear the CREN bit.
3. If interrupts are desired, then set the TXIE bit.
4. If 9-bit transmission is desired, then set the TX9 bit.
5. Start transmission by loading data to TXREG.
6. If 9-bit transmission is selected, the ninth bit should be loaded in TX9D.
7. Enable the transmission by setting TXEN.

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

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

### 13.4.2 USART SYNCHRONOUS SLAVE RECEPTION

Operation of the synchronous master and slave modes are identical except in the case of the SLEEP mode. Also, SREN is a don't care in slave mode.

If receive is enabled (CREN) prior to the SLEEP instruction, then a word may be received during SLEEP. On completely receiving the word, the RSR will transfer the data to RCREG (setting RCIF) and if the RCIE bit is set, the interrupt generated will wake the chip from SLEEP. If the global interrupt is enabled, the program will branch to the interrupt vector (0020h).

Steps to follow when setting up a Synchronous Slave Reception:

1. Enable the synchronous master serial port by setting the SYNC and SPEN bits and clearing the CSRC bit.
2. If interrupts are desired, then set the RCIE bit.
3. If 9-bit reception is desired, then set the RX9 bit.
4. To enable reception, set the CREN bit.
5. The RCIF bit will be set when reception is complete and an interrupt will be generated if the RCIE bit was set.
6. Read RCSTA to get the ninth bit (if enabled) and determine if any error occurred during reception.
7. Read the 8-bit received data by reading RCREG.
8. If any error occurred, clear the error by clearing the CREN bit.

**Note:** To abort reception, either clear the SPEN bit, the SREN bit (when in single receive mode), or the CREN bit (when in continuous receive mode). This will reset the receive logic, so that it will be in the proper state when receive is re-enabled.

# PIC17C4X

**TABLE 13-9: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE 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	TX7	TX6	TX5	TX4	TX3	TX2	TX1	TX0	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 synchronous slave transmission.

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

**TABLE 13-10: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE RECEPTION**

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, Bank1	PIR	RBIF	TMR3IF	TMR2IF	TMR1IF	CA2IF	CA1IF	TXIF	RCIF	0000 0010	0000 0010
13h, Bank0	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00u
14h, Bank0	RCREG	RX7	RX6	RX5	RX4	RX3	RX2	RX1	RX0	xxxx xxxx	uuuu uuuu
17h, Bank1	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, Bank0	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 synchronous slave reception.

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

## 14.0 SPECIAL FEATURES OF THE CPU

What sets a microcontroller apart from other processors are special circuits to deal with the needs of real time applications. The PIC17CXX family has a host of such features intended to maximize system reliability, minimize cost through elimination of external components, provide power saving operating modes and offer code protection. These are:

- OSC selection
- Reset
  - Power-on Reset (POR)
  - Power-up Timer (PWRT)
  - Oscillator Start-up Timer (OST)
- Interrupts
- Watchdog Timer (WDT)
- SLEEP
- Code protection

The PIC17CXX has a Watchdog Timer which can be shut off only through EPROM bits. It runs off its own RC oscillator for added reliability. There are two timers that offer necessary delays on power-up. One is the Oscillator Start-up Timer (OST), intended to keep the chip in RESET until the crystal oscillator is stable. The other is the Power-up Timer (PWRT), which provides a fixed delay of 96 ms (nominal) on power-up only, designed to keep the part in RESET while the power supply stabilizes. With these two timers on-chip, most applications need no external reset circuitry.

The SLEEP mode is designed to offer a very low current power-down mode. The user can wake from SLEEP through external reset, Watchdog Timer Reset or through an interrupt. Several oscillator options are also made available to allow the part to fit the application. The RC oscillator option saves system cost while the LF crystal option saves power. Configuration bits are used to select various options. This configuration word has the format shown in Figure 14-1.

**FIGURE 14-1: CONFIGURATION WORD**



# PIC17C4X

## 14.1 Configuration Bits

The PIC17CXX has up to seven configuration locations (Table 14-1). These locations can be programmed (read as '0') or left unprogrammed (read as '1') to select various device configurations. Any write to a configuration location, regardless of the data, will program that configuration bit. A `TABLWT` instruction is required to write to program memory locations. The configuration bits can be read by using the `TABLRD` instructions. Reading any configuration location between FE00h and FE07h will read the low byte of the configuration word (Figure 14-1) into the `TABLATL` register. The `TABLATH` register will be FFh. Reading a configuration location between FE08h and FE0Fh will read the high byte of the configuration word into the `TABLATL` register. The `TABLATH` register will be FFh.

Addresses FE00h through FE0Fh are only in the program memory space for microcontroller and code protected microcontroller modes. A device programmer will be able to read the configuration word in any processor mode. See programming specifications for more detail.

**TABLE 14-1: CONFIGURATION LOCATIONS**

Bit	Address
FOSC0	FE00h
FOSC1	FE01h
WDTPS0	FE02h
WDTPS1	FE03h
PM0	FE04h
PM1	FE06h
PM2 <sup>(1)</sup>	FE0Fh <sup>(1)</sup>

Note 1: This location does not exist on the PIC17C42.

**Note:** When programming the desired configuration locations, they must be programmed in ascending order. Starting with address FE00h.

## 14.2 Oscillator Configurations

### 14.2.1 OSCILLATOR TYPES

The PIC17CXX can be operated in four different oscillator modes. The user can program two configuration bits (FOSC1:FOSC0) to select one of these four modes:

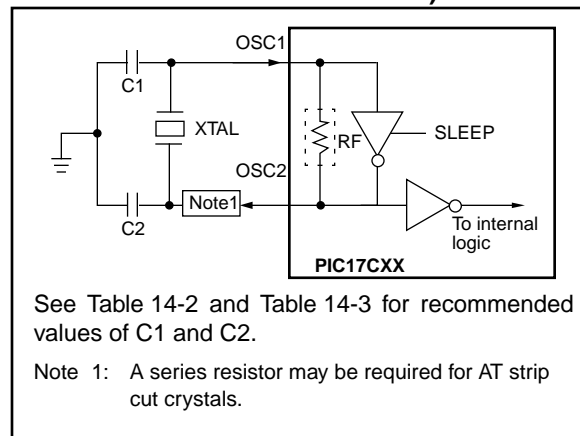
- LF: Low Power Crystal
- XT: Crystal/Resonator
- EC: External Clock Input
- RC: Resistor/Capacitor

### 14.2.2 CRYSTAL OSCILLATOR / CERAMIC RESONATORS

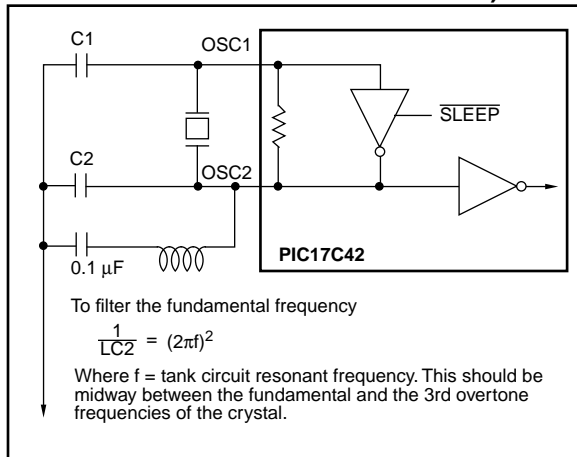
In XT or LF modes, a crystal or ceramic resonator is connected to the `OSC1/CLKIN` and `OSC2/CLKOUT` pins to establish oscillation (Figure 14-2). The PIC17CXX Oscillator design requires the use of a parallel cut crystal. Use of a series cut crystal may give a frequency out of the crystal manufacturers specifications.

For frequencies above 20 MHz, it is common for the crystal to be an overtone mode crystal. Use of overtone mode crystals require a tank circuit to attenuate the gain at the fundamental frequency. Figure 14-3 shows an example of this.

**FIGURE 14-2: CRYSTAL OR CERAMIC RESONATOR OPERATION (XT OR LF OSC CONFIGURATION)**



**FIGURE 14-3: CRYSTAL OPERATION, OVERTONE CRYSTALS (XT OSC CONFIGURATION)**



**TABLE 14-2: CAPACITOR SELECTION FOR CERAMIC RESONATORS**

Oscillator Type	Resonator Frequency	Capacitor Range C1 = C2
LF	455 kHz	15 - 68 pF
	2.0 MHz	10 - 33 pF
XT	4.0 MHz	22 - 68 pF
	8.0 MHz	33 - 100 pF
	16.0 MHz	33 - 100 pF

Higher capacitance increases the stability of the oscillator but also increases the start-up time. These values are for design guidance only. Since each resonator has its own characteristics, the user should consult the resonator manufacturer for appropriate values of external components.

**Resonators Used:**

455 kHz	Panasonic EFO-A455K04B	± 0.3%
2.0 MHz	Murata Erie CSA2.00MG	± 0.5%
4.0 MHz	Murata Erie CSA4.00MG	± 0.5%
8.0 MHz	Murata Erie CSA8.00MT	± 0.5%
16.0 MHz	Murata Erie CSA16.00MX	± 0.5%

Resonators used did not have built-in capacitors.

**TABLE 14-3: CAPACITOR SELECTION FOR CRYSTAL OSCILLATOR**

Osc Type	Freq	C1	C2
LF	32 kHz <sup>(1)</sup>	100-150 pF	100-150 pF
	1 MHz	10-33 pF	10-33 pF
	2 MHz	10-33 pF	10-33 pF
XT	2 MHz	47-100 pF	47-100 pF
	4 MHz	15-68 pF	15-68 pF
	8 MHz <sup>(2)</sup>	15-47 pF	15-47 pF
	16 MHz	TBD	TBD
	25 MHz	15-47 pF	15-47 pF
	32 MHz <sup>(3)</sup>	0 <sup>(3)</sup>	0 <sup>(3)</sup>

Higher capacitance increases the stability of the oscillator but also increases the start-up time and the oscillator current. These values are for design guidance only. Rs may be required in XT mode to avoid overdriving the crystals with low drive level specification. Since each crystal has its own characteristics, the user should consult the crystal manufacturer for appropriate values for external components.

Note 1: For VDD > 4.5V, C1 = C2 ≈ 30 pF is recommended.

2: Rs of 330Ω is required for a capacitor combination of 15/15 pF.

3: Only the capacitance of the board was present.

**Crystals Used:**

32.768 kHz	Epson C-001R32.768K-A	± 20 PPM
1.0 MHz	ECS-10-13-1	± 50 PPM
2.0 MHz	ECS-20-20-1	± 50 PPM
4.0 MHz	ECS-40-20-1	± 50 PPM
8.0 MHz	ECS ECS-80-S-4 ECS-80-18-1	± 50 PPM
16.0 MHz	ECS-160-20-1	TBD
25 MHz	CTS CTS25M	± 50 PPM
32 MHz	CRYSTEK HF-2	± 50 PPM

### 14.2.3 EXTERNAL CLOCK OSCILLATOR

In the EC oscillator mode, the OSC1 input can be driven by CMOS drivers. In this mode, the OSC1/CLKIN pin is hi-impedance and the OSC2/CLKOUT pin is the CLKOUT output (4 TOSC).

**FIGURE 14-4: EXTERNAL CLOCK INPUT OPERATION (EC OSC CONFIGURATION)**



# PIC17C4X

## 14.2.4 EXTERNAL CRYSTAL OSCILLATOR CIRCUIT

Either a prepackaged oscillator can be used or a simple oscillator circuit with TTL gates can be built. Prepackaged oscillators provide a wide operating range and better stability. A well-designed crystal oscillator will provide good performance with TTL gates. Two types of crystal oscillator circuits can be used: one with series resonance, or one with parallel resonance.

Figure 14-5 shows implementation of a parallel resonant oscillator circuit. The circuit is designed to use the fundamental frequency of the crystal. The 74AS04 inverter performs the 180-degree phase shift that a parallel oscillator requires. The 4.7 kΩ resistor provides the negative feedback for stability. The 10 kΩ potentiometer biases the 74AS04 in the linear region. This could be used for external oscillator designs.

**FIGURE 14-5: EXTERNAL PARALLEL RESONANT CRYSTAL OSCILLATOR CIRCUIT**



Figure 14-6 shows a series resonant oscillator circuit. This circuit is also designed to use the fundamental frequency of the crystal. The inverter performs a 180-degree phase shift in a series resonant oscillator circuit. The 330 kΩ resistors provide the negative feedback to bias the inverters in their linear region.

**FIGURE 14-6: EXTERNAL SERIES RESONANT CRYSTAL OSCILLATOR CIRCUIT**



## 14.2.5 RC OSCILLATOR

For timing insensitive applications, the RC device option offers additional cost savings. RC oscillator frequency is a function of the supply voltage, the resistor ( $R_{ext}$ ) and capacitor ( $C_{ext}$ ) values, and the operating temperature. In addition to this, oscillator frequency will vary from unit to unit due to normal process parameter variation. Furthermore, the difference in lead frame capacitance between package types will also affect oscillation frequency, especially for low  $C_{ext}$  values. The user also needs to take into account variation due to tolerance of external R and C components used. Figure 14-6 shows how the R/C combination is connected to the PIC17CXX. For  $R_{ext}$  values below 2.2 kΩ, the oscillator operation may become unstable, or stop completely. For very high  $R_{ext}$  values (e.g. 1 MΩ), the oscillator becomes sensitive to noise, humidity and leakage. Thus, we recommend to keep  $R_{ext}$  between 3 kΩ and 100 kΩ.

Although the oscillator will operate with no external capacitor ( $C_{ext} = 0$  pF), we recommend using values above 20 pF for noise and stability reasons. With little or no external capacitance, oscillation frequency can vary dramatically due to changes in external capacitances, such as PCB trace capacitance or package lead frame capacitance.

See Section 18.0 for RC frequency variation from part to part due to normal process variation. The variation is larger for larger R (since leakage current variation will affect RC frequency more for large R) and for smaller C (since variation of input capacitance will affect RC frequency more).

See Section 18.0 for variation of oscillator frequency due to  $V_{DD}$  for given  $R_{ext}/C_{ext}$  values as well as frequency variation due to operating temperature for given R, C, and  $V_{DD}$  values.

The oscillator frequency, divided by 4, is available on the OSC2/CLKOUT pin, and can be used for test purposes or to synchronize other logic (see Figure 3-2 for waveform).

**FIGURE 14-7: RC OSCILLATOR MODE**



## 14.3 Watchdog Timer (WDT)

The Watchdog Timer's function is to recover from software malfunction. The WDT uses an internal free running on-chip RC oscillator for its clock source. This does not require any external components. This RC oscillator is separate from the RC oscillator of the OSC1/CLKIN pin. That means that the WDT will run, even if the clock on the OSC1/CLKIN and OSC2/CLKOUT pins of the device has been stopped, for example, by execution of a SLEEP instruction. During normal operation and SLEEP mode, a WDT time-out generates a device RESET. The WDT can be permanently disabled by programming the configuration bits WDTPS1:WDTPS0 as '00' (Section 14.1).

Under normal operation, the WDT must be cleared on a regular interval. This time is less the minimum WDT overflow time. Not clearing the WDT in this time frame will cause the WDT to overflow and reset the device.

### 14.3.1 WDT PERIOD

The WDT has a nominal time-out period of 12 ms, (with postscaler = 1). The time-out periods vary with temperature, VDD and process variations from part to part (see DC specs). If longer time-out periods are desired, a postscaler with a division ratio of up to 1:256 can be assigned to the WDT. Thus, typical time-out periods up to 3.0 seconds can be realized.

The CLRWDT and SLEEP instructions clear the WDT and the postscaler (if assigned to the WDT) and prevent it from timing out thus generating a device RESET condition.

The  $\overline{TO}$  bit in the CPUTSTA register will be cleared upon a WDT time-out.

### 14.3.2 CLEARING THE WDT AND POSTSCALER

The WDT and postscaler are cleared when:

- The device is in the reset state
- A SLEEP instruction is executed
- A CLRWDT instruction is executed
- Wake-up from SLEEP by an interrupt

The WDT counter/postscaler will start counting on the first edge after the device exits the reset state.

### 14.3.3 WDT PROGRAMMING CONSIDERATIONS

It should also be taken in account that under worst case conditions (VDD = Min., Temperature = Max., max. WDT postscaler) it may take several seconds before a WDT time-out occurs.

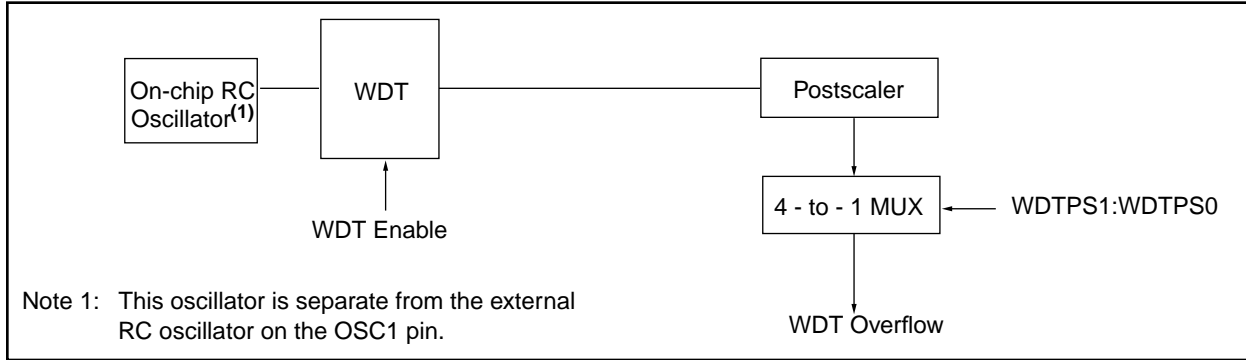
The WDT and postscaler is the Power-up Timer during the Power-on Reset sequence.

### 14.3.4 WDT AS NORMAL TIMER

When the WDT is selected as a normal timer, the clock source is the device clock. Neither the WDT nor the postscaler are directly readable or writable. The overflow time is 65536 TOSC cycles. On overflow, the  $\overline{TO}$  bit is cleared (device is not reset). The CLRWDT instruction can be used to set the  $\overline{TO}$  bit. This allows the WDT to be a simple overflow timer. When in sleep, the WDT does not increment.

# PIC17C4X

**FIGURE 14-8: WATCHDOG TIMER BLOCK DIAGRAM**



**TABLE 14-4: REGISTERS/BITS ASSOCIATED WITH THE WATCHDOG TIMER**

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)
—	Config	—	PM1	—	PM0	WDTPS1	WDTPS0	FOSC1	FOSC0	(Note 2)	(Note 2)
06h, Unbanked	CPUSTA	—	—	STKAV	GLINTD	$\overline{T0}$	$\overline{PD}$	—	—	--11 11--	--11 qq--

Legend: — = unimplemented read as '0', q - value depends on condition, shaded cells are not used by the WDT.

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

2: This value will be as the device was programmed, or if unprogrammed, will read as all '1's.



## 14.4 Power-down Mode (SLEEP)

The Power-down mode is entered by executing a SLEEP instruction. This clears the Watchdog Timer and postscaler (if enabled). The  $\overline{PD}$  bit is cleared and the  $\overline{TO}$  bit is set (in the CPUSTA register). In SLEEP mode, the oscillator driver is turned off. The I/O ports maintain their status (driving high, low, or hi-impedance).

The  $\overline{MCLR}/VPP$  pin must be at a logic high level ( $V_{IHMC}$ ). A WDT time-out RESET does not drive the  $\overline{MCLR}/VPP$  pin low.

### 14.4.1 WAKE-UP FROM SLEEP

The device can wake up from SLEEP through one of the following events:

- A POR reset
- External reset input on  $\overline{MCLR}/VPP$  pin
- WDT Reset (if WDT was enabled)
- Interrupt from RA0/INT pin, RB port change, TOCKI interrupt, or some Peripheral Interrupts

The following peripheral interrupts can wake-up from SLEEP:

- Capture1 interrupt
- Capture2 interrupt
- USART synchronous slave transmit interrupt
- USART synchronous slave receive interrupt

Other peripherals can not generate interrupts since during SLEEP, no on-chip Q clocks are present.

Any reset event will cause a device reset. Any interrupt event is considered a continuation of program execution. The  $\overline{TO}$  and  $\overline{PD}$  bits in the CPUSTA register can be used to determine the cause of device reset. The

$\overline{PD}$  bit, which is set on power-up, is cleared when SLEEP is invoked. The  $\overline{TO}$  bit is cleared if WDT time-out occurred (and caused wake-up).

When the SLEEP instruction is being executed, the next instruction (PC + 1) is pre-fetched. For the device to wake-up through an interrupt event, the corresponding interrupt enable bit must be set (enabled). Wake-up is regardless of the state of the GLINTD bit. If the GLINTD bit is set (disabled), the device continues execution at the instruction after the SLEEP instruction. If the GLINTD bit is clear (enabled), the device executes the instruction after the SLEEP instruction and then branches to the interrupt vector address. In cases where the execution of the instruction following SLEEP is not desirable, the user should have a NOP after the SLEEP instruction.

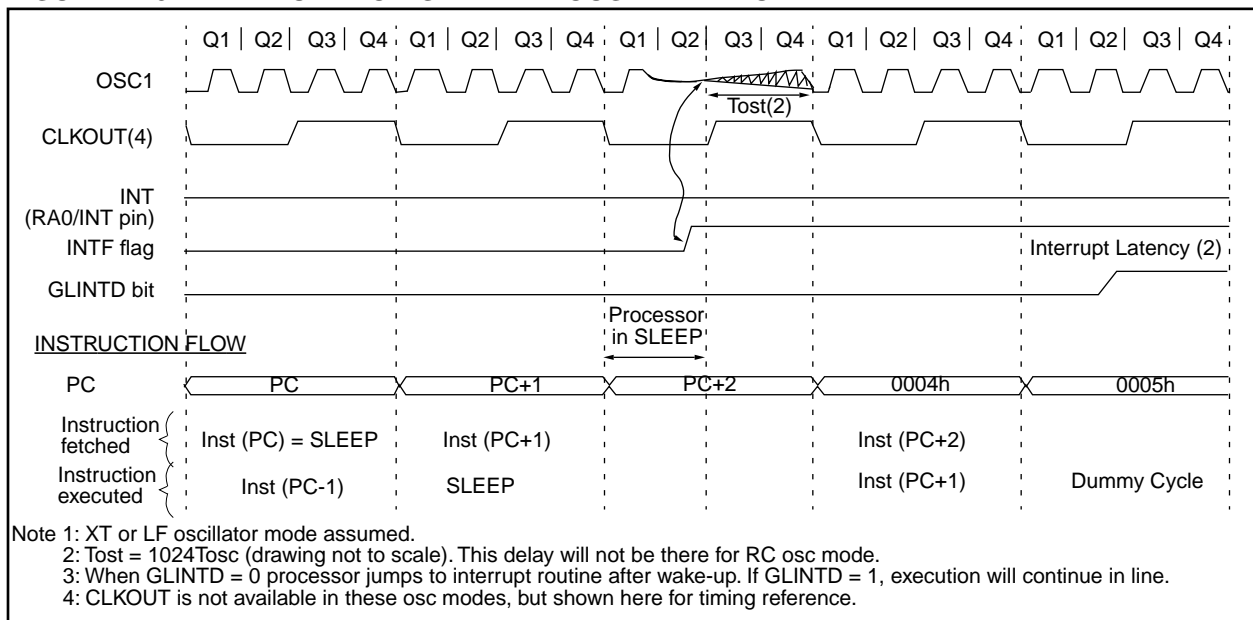
**Note:** If the global interrupts are disabled (GLINTD is set), but any interrupt source has both its interrupt enable bit and the corresponding interrupt flag bits set, the device will immediately wake-up from sleep. The  $\overline{TO}$  bit is set, and the  $\overline{PD}$  bit is cleared.

The WDT is cleared when the device wake from SLEEP, regardless of the source of wake-up.

#### 14.4.1.1 WAKE-UP DELAY

When the oscillator type is configured in XT or LF mode, the Oscillator Start-up Timer (OST) is activated on wake-up. The OST will keep the device in reset for  $1024T_{osc}$ . This needs to be taken into account when considering the interrupt response time when coming out of SLEEP.

**FIGURE 14-9: WAKE-UP FROM SLEEP THROUGH INTERRUPT**



## 14.4.2 MINIMIZING CURRENT CONSUMPTION

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

## 14.5 Code Protection

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

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

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

**Note:** Microchip does not recommend code protecting windowed devices.

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

## 15.0 INSTRUCTION SET SUMMARY

The PIC17CXX instruction set consists of 58 instructions. Each instruction is a 16-bit word divided into an OPCODE and one or more operands. The opcode specifies the instruction type, while the operand(s) further specify the operation of the instruction. The PIC17CXX instruction set can be grouped into three types:

- byte-oriented
- bit-oriented
- literal and control operations.

These formats are shown in Figure 15-1.

Table 15-1 shows the field descriptions for the opcodes. These descriptions are useful for understanding the opcodes in Table 15-2 and in each specific instruction descriptions.

**byte-oriented instructions**, 'f' represents a file register designator and 'd' represents a destination designator. The file register designator specifies which file register is to be used by the instruction.

The destination designator specifies where the result of the operation is to be placed. If 'd' = '0', the result is placed in the WREG register. If 'd' = '1', the result is placed in the file register specified by the instruction.

**bit-oriented instructions**, 'b' represents a bit field designator which selects the number of the bit affected by the operation, while 'f' represents the number of the file in which the bit is located.

**literal and control operations**, 'k' represents an 8- or 11-bit constant or literal value.

The instruction set is highly orthogonal and is grouped into:

- byte-oriented operations
- bit-oriented operations
- literal and control operations

All instructions are executed within one single instruction cycle, unless:

- a conditional test is true
- the program counter is changed as a result of an instruction
- a table read or a table write instruction is executed (in this case, the execution takes two instruction cycles with the second cycle executed as a NOP)

One instruction cycle consists of four oscillator periods. Thus, for an oscillator frequency of 25 MHz, the normal instruction execution time is 160 ns. If a conditional test is true or the program counter is changed as a result of an instruction, the instruction execution time is 320 ns.

**TABLE 15-1: OPCODE FIELD DESCRIPTIONS**

Field	Description
f	Register file address (00h to FFh)
p	Peripheral register file address (00h to 1Fh)
i	Table pointer control i = '0' (do not change) i = '1' (increment after instruction execution)
t	Table byte select t = '0' (perform operation on lower byte) t = '1' (perform operation on upper byte literal field, constant data)
WREG	Working register (accumulator)
b	Bit address within an 8-bit file register
k	Literal field, constant data or label
x	Don't care location (= '0' or '1') The assembler will generate code with x = '0'. It is the recommended form of use for compatibility with all Microchip software tools.
d	Destination select 0 = store result in WREG 1 = store result in file register f Default is d = '1'
u	Unused, encoded as '0'
s	Destination select 0 = store result in file register f and in the WREG 1 = store result in file register f Default is s = '1'
label	Label name
C, DC, Z, OV	ALU status bits Carry, Digit Carry, Zero, Overflow
GLINTD	Global Interrupt Disable bit (CPUSTA<4>)
TBLPTR	Table Pointer (16-bit)
TBLAT	Table Latch (16-bit) consists of high byte (TBLATH) and low byte (TBLATL)
TBLATL	Table Latch low byte
TBLATH	Table Latch high byte
TOS	Top of Stack
PC	Program Counter
BSR	Bank Select Register
WDT	Watchdog Timer Counter
$\overline{TO}$	Time-out bit
$\overline{PD}$	Power-down bit
dest	Destination either the WREG register or the specified register file location
[ ]	Options
( )	Contents
→	Assigned to
< >	Register bit field
∈	In the set of
<i>italics</i>	User defined term (font is courier)

# PIC17C4X

Table 15-2 lists the instructions recognized by the MPASM assembler.

**Note 1:** Any unused opcode is Reserved. Use of any reserved opcode may cause unexpected operation.

**Note 2:** The shaded instructions are not available in the PIC17C42

All instruction examples use the following format to represent a hexadecimal number:

0xhh

where h signifies a hexadecimal digit.

To represent a binary number:

0000 0100b

where b signifies a binary string.

**FIGURE 15-1: GENERAL FORMAT FOR INSTRUCTIONS**



## 15.1 Special Function Registers as Source/Destination

The PIC17C4X's orthogonal instruction set allows read and write of all file registers, including special function registers. There are some special situations the user should be aware of:

### 15.1.1 ALUSTA AS DESTINATION

If an instruction writes to ALUSTA, the Z, C, DC and OV bits may be set or cleared as a result of the instruction and overwrite the original data bits written. For example, executing `CLRF ALUSTA` will clear register ALUSTA, and then set the Z bit leaving `0000 0100b` in the register.

### 15.1.2 PCL AS SOURCE OR DESTINATION

Read, write or read-modify-write on PCL may have the following results:

Read PC: PCH → PCLATH; PCL → dest

Write PCL: PCLATH → PCH;  
8-bit destination value → PCL

Read-Modify-Write: PCL → ALU operand  
PCLATH → PCH;  
8-bit result → PCL

Where PCH = program counter high byte (not an addressable register), PCLATH = Program counter high holding latch, dest = destination, WREG or f.

### 15.1.3 BIT MANIPULATION

All bit manipulation instructions are done by first reading the entire register, operating on the selected bit and writing the result back (read-modify-write). The user should keep this in mind when operating on special function registers, such as ports.

## 15.2 Q Cycle Activity

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

The 4 Q cycles that make up an instruction cycle ( $T_{cy}$ ) can be generalized as:

Q1: Instruction Decode Cycle or forced NOP

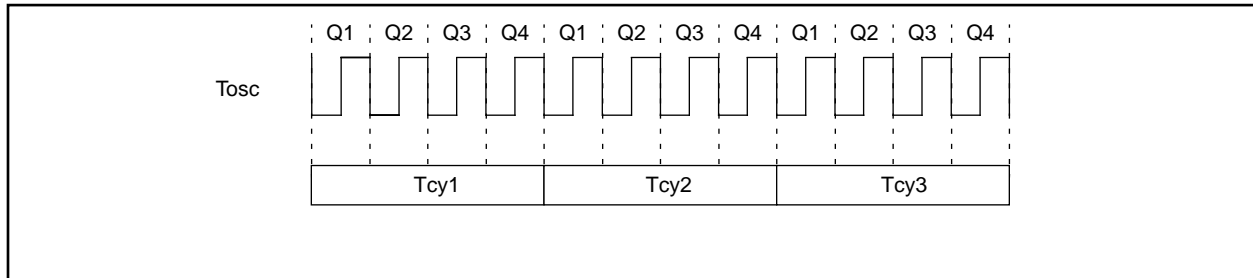
Q2: Instruction Read Cycle or NOP

Q3: Instruction Execute

Q4: Instruction Write Cycle or NOP

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

**FIGURE 15-2: Q CYCLE ACTIVITY**



# PIC17C4X

**TABLE 15-2: PIC17CXX INSTRUCTION SET**

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

**TABLE 15-2: PIC17CXX INSTRUCTION SET (Cont.'d)**

Mnemonic, Operands	Description	Cycles	16-bit Opcode		Status Affected	Notes
			MSb	LSb		
<b>TABLWT</b> t,i,f	Table Write	2	1010	11ti ffff ffff	None	5
<b>TLRD</b> t,f	Table Latch Read	1	1010	00tx ffff ffff	None	
<b>TLWT</b> t,f	Table Latch Write	1	1010	01tx ffff ffff	None	
<b>TSTFSZ</b> f	Test f, skip if 0	1 (2)	0011	0011 ffff ffff	None	6,8
<b>XORWF</b> f,d	Exclusive OR WREG with f	1	0000	110d ffff ffff	Z	
<b>BIT-ORIENTED FILE REGISTER OPERATIONS</b>						
<b>BCF</b> f,b	Bit Clear f	1	1000	1bbb ffff ffff	None	
<b>BSF</b> f,b	Bit Set f	1	1000	0bbb ffff ffff	None	
<b>BTFSC</b> f,b	Bit test, skip if clear	1 (2)	1001	1bbb ffff ffff	None	6,8
<b>BTFSS</b> f,b	Bit test, skip if set	1 (2)	1001	0bbb ffff ffff	None	6,8
<b>BTG</b> f,b	Bit Toggle f	1	0011	1bbb ffff ffff	None	
<b>LITERAL AND CONTROL OPERATIONS</b>						
<b>ADDLW</b> k	ADD literal to WREG	1	1011	0001 kkkk kkkk	OV,C,DC,Z	
<b>ANDLW</b> k	AND literal with WREG	1	1011	0101 kkkk kkkk	Z	
<b>CALL</b> k	Subroutine Call	2	111k	kkkk kkkk kkkk	None	7
<b>CLRWDT</b> —	Clear Watchdog Timer	1	0000	0000 0000 0100	$\overline{TO}, \overline{PD}$	
<b>GOTO</b> k	Unconditional Branch	2	110k	kkkk kkkk kkkk	None	7
<b>IORLW</b> k	Inclusive OR literal with WREG	1	1011	0011 kkkk kkkk	Z	
<b>LCALL</b> k	Long Call	2	1011	0111 kkkk kkkk	None	4,7
<b>MOVLB</b> k	Move literal to low nibble in BSR	1	1011	1000 uuuu kkkk	None	
<b>MOVLR</b> k	Move literal to high nibble in BSR	1	1011	101x kkkk uuuu	None	9
<b>MOVLW</b> k	Move literal to WREG	1	1011	0000 kkkk kkkk	None	
<b>MULLW</b> k	Multiply literal with WREG	1	1011	1100 kkkk kkkk	None	9
<b>RETFIE</b> —	Return from interrupt (and enable interrupts)	2	0000	0000 0000 0101	GLINTD	7
<b>RETLW</b> k	Return literal to WREG	2	1011	0110 kkkk kkkk	None	7
<b>RETURN</b> —	Return from subroutine	2	0000	0000 0000 0010	None	7
<b>SLEEP</b> —	Enter SLEEP Mode	1	0000	0000 0000 0011	$\overline{TO}, \overline{PD}$	
<b>SUBLW</b> k	Subtract WREG from literal	1	1011	0010 kkkk kkkk	OV,C,DC,Z	
<b>XORLW</b> k	Exclusive OR literal with WREG	1	1011	0100 kkkk kkkk	Z	

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.

# PIC17C4X

## ADDLW      ADD Literal to WREG

Syntax:            [ *label* ] ADDLW   k  
 Operands:         $0 \leq k \leq 255$   
 Operation:        (WREG) + k  $\rightarrow$  (WREG)  
 Status Affected:  OV, C, DC, Z  
 Encoding:        

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

  
 Description:     The contents of WREG are added to the 8-bit literal 'k' and the result is placed in WREG.  
 Words:            1  
 Cycles:           1

Q Cycle Activity:

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

Example:            ADDLW   0x15

Before Instruction  
       WREG = 0x10  
 After Instruction  
       WREG = 0x25

## ADDWF      ADD WREG to f

Syntax:            [ *label* ] ADDWF   f,d  
 Operands:         $0 \leq f \leq 255$   
                        $d \in [0,1]$   
 Operation:        (WREG) + (f)  $\rightarrow$  (dest)  
 Status Affected:  OV, C, DC, Z  
 Encoding:        

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

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

Q Cycle Activity:

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

Example:            ADDWF   REG, 0

Before Instruction  
       WREG = 0x17  
       REG   = 0xC2  
 After Instruction  
       WREG = 0xD9  
       REG   = 0xC2



## ADDWFC      ADD WREG and Carry bit to f

**Syntax:**            [ *label* ] ADDWFC    f,d

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

**Operation:**        (WREG) + (f) + C → (dest)

**Status Affected:**  OV, C, DC, Z

**Encoding:**

0001	000d	ffff	ffff
------	------	------	------

**Description:**     Add WREG, the Carry Flag and data memory location 'f'. If 'd' is 0, the result is placed in WREG. If 'd' is 1, the 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 to destination

**Example:**            ADDWFC      REG    0

**Before Instruction**

Carry bit = 1  
 REG = 0x02  
 WREG = 0x4D

**After Instruction**

Carry bit = 0  
 REG = 0x02  
 WREG = 0x50

## ANDLW      And Literal with WREG

**Syntax:**            [ *label* ] ANDLW    k

**Operands:**         $0 \leq k \leq 255$

**Operation:**        (WREG) .AND. (k) → (WREG)

**Status Affected:**  Z

**Encoding:**

1011	0101	kkkk	kkkk
------	------	------	------

**Description:**     The contents of WREG are AND'ed with the 8-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:**            ANDLW      0x5F

**Before Instruction**

WREG = 0xA3

**After Instruction**

WREG = 0x03

# PIC17C4X

## ANDWF AND WREG with f

**Syntax:** [ *label* ] ANDWF f,d

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

**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:**  $0 \leq f \leq 255$   
 $0 \leq b \leq 7$

**Operation:**  $0 \rightarrow (f<b>)$

**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

**BSF**                      **Bit Set f**

---

Syntax:                    [ *label* ] BSF f,b

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

Operation:                 $1 \rightarrow (f<b>)$

Status Affected:        None

Encoding:                

1000	0bbb	ffff	ffff
------	------	------	------

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

Words:                    1

Cycles:                    1

Q Cycle Activity:

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

Example:                BSF        FLAG\_REG, 7

Before Instruction  
FLAG\_REG= 0x0A

After Instruction  
FLAG\_REG= 0x8A

**BTFSC**                    **Bit Test, skip if Clear**

---

Syntax:                    [ *label* ] BTFSC f,b

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

Operation:                skip if  $(f<b>) = 0$

Status Affected:        None

Encoding:                

1001	1bbb	ffff	ffff
------	------	------	------

Description:             If bit 'b' in register 'f' is 0 then the next instruction is skipped.  
If bit 'b' is 0 then the next instruction fetched during the current instruction execution is discarded, and a 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

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    BTFSC    FLAG, 1  
                              FALSE    :  
                              TRUE     :

Before Instruction  
PC = address (HERE)

After Instruction  
If FLAG<1> = 0;  
PC = address (TRUE)  
If FLAG<1> = 1;  
PC = address (FALSE)

# PIC17C4X

## **BTFS** Bit Test, skip if Set

Syntax: [ *label* ] BTFS *f*,*b*

Operands:  $0 \leq f \leq 127$   
 $0 \leq b < 7$

Operation: skip if (*f*<*b*>) = 1

Status Affected: None

Encoding: 

1001	0bbb	ffff	ffff
------	------	------	------

Description: If bit 'b' in register 'f' is 1 then the next instruction is skipped.  
 If bit 'b' is 1, then the next instruction fetched during the current instruction execution, 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   BTFS   FLAG, 1  
                       FALSE  :  
                       TRUE   :

Before Instruction

PC = address (HERE)

After Instruction

If FLAG<1> = 0;  
 PC = address (FALSE)  
 If FLAG<1> = 1;  
 PC = address (TRUE)

## **BTG** Bit Toggle f

Syntax: [ *label* ] BTG *f*,*b*

Operands:  $0 \leq f \leq 255$   
 $0 \leq b < 7$

Operation: ( $\overline{f\langle b \rangle}$ )  $\rightarrow$  (*f*<*b*>)

Status Affected: None

Encoding: 

0011	1bbb	ffff	ffff
------	------	------	------

Description: Bit 'b' in data memory location 'f' is inverted.

Words: 1

Cycles: 1

Q Cycle Activity:

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

**Example:**           BTG   PORTC, 4

Before Instruction:

PORTC = 0111 0101 [0x75]

After Instruction:

PORTC = 0110 0101 [0x65]

## CALL Subroutine Call

**Syntax:** `[label] CALL k`

**Operands:**  $0 \leq k \leq 4095$

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

**Status Affected:** None

**Encoding:**

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

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

**Words:** 1

**Cycles:** 2

**Q Cycle Activity:**

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

**Example:**           HERE       CALL   THERE

**Before Instruction**

PC = Address (HERE)

**After Instruction**

PC = Address (THERE)

TOS = Address (HERE + 1)

## CLRF Clear f

**Syntax:** `[label] CLRF f,s`

**Operands:**  $0 \leq f \leq 255$

**Operation:**  $00h \rightarrow f$ ,  $s \in [0,1]$   
 $00h \rightarrow dest$

**Status Affected:** None

**Encoding:**

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

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

**Words:** 1

**Cycles:** 1

**Q Cycle Activity:**

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

**Example:**                   CLRF       FLAG\_REG

**Before Instruction**

FLAG\_REG = 0x5A

**After Instruction**

FLAG\_REG = 0x00

# PIC17C4X

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

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

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

## CPFSEQ Compare f with WREG, skip if f = WREG

**Syntax:** [ *label* ] CPFSEQ f

**Operands:**  $0 \leq f \leq 255$

**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:**  $0 \leq f \leq 255$

**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

## CPFSLT **Compare f with WREG, skip if f < WREG**

**Syntax:** [label] CPFSLT f

**Operands:**  $0 \leq f \leq 255$

**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:**  $0 \leq f \leq 255$   
 $s \in [0,1]$

**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

**Example 1:** 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
```



DECF	Decrement f								
Syntax:	[ <i>label</i> ] DECF f,d								
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$								
Operation:	$(f) - 1 \rightarrow (\text{dest})$								
Status Affected:	OV, C, DC, Z								
Encoding:	<table border="1"> <tr> <td>0000</td> <td>011d</td> <td>ffff</td> <td>ffff</td> </tr> </table>	0000	011d	ffff	ffff				
0000	011d	ffff	ffff						
Description:	Decrement 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:	<table border="1"> <thead> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> </thead> <tbody> <tr> <td>Decode</td> <td>Read register 'f'</td> <td>Execute</td> <td>Write to destination</td> </tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Execute	Write to destination
Q1	Q2	Q3	Q4						
Decode	Read register 'f'	Execute	Write to destination						

**Example:** DECF CNT, 1

Before Instruction

CNT = 0x01  
Z = 0

After Instruction

CNT = 0x00  
Z = 1

DECFSZ	Decrement f, skip if 0								
Syntax:	[ <i>label</i> ] DECFSZ f,d								
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$								
Operation:	$(f) - 1 \rightarrow (\text{dest});$ skip if result = 0								
Status Affected:	None								
Encoding:	<table border="1"> <tr> <td>0001</td> <td>011d</td> <td>ffff</td> <td>ffff</td> </tr> </table>	0001	011d	ffff	ffff				
0001	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 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:	<table border="1"> <thead> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> </thead> <tbody> <tr> <td>Decode</td> <td>Read register 'f'</td> <td>Execute</td> <td>Write to destination</td> </tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Execute	Write to destination
Q1	Q2	Q3	Q4						
Decode	Read register 'f'	Execute	Write to destination						

**Example:**            HERE            DECFSZ CNT, 1  
  GOTO LOOP  
  CONTINUE

Before Instruction

PC = Address (HERE)

After Instruction

CNT = CNT - 1  
If CNT = 0;  
    PC = Address (CONTINUE)  
If CNT  $\neq$  0;  
    PC = Address (HERE+1)

# 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   ≠    0;  
     PC            =    Address ( NZERO )

## GOTO            Unconditional Branch

Syntax:            *[label]* GOTO k

Operands:         $0 \leq k \leq 8191$

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

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 )

<b>INCF</b>	<b>Increment f</b>								
Syntax:	[ <i>label</i> ] INCF f,d								
Operands:	0 ≤ f ≤ 255 d ∈ [0,1]								
Operation:	(f) + 1 → (dest)								
Status Affected:	OV, C, DC, Z								
Encoding:	<table border="1" style="display: inline-table;"><tr><td>0001</td><td>010d</td><td>ffff</td><td>ffff</td></tr></table>	0001	010d	ffff	ffff				
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:	<table border="1" style="display: inline-table;"><thead><tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr></thead><tbody><tr><td>Decode</td><td>Read register 'f'</td><td>Execute</td><td>Write to destination</td></tr></tbody></table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Execute	Write to destination
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

<b>INCFSZ</b>	<b>Increment f, skip if 0</b>								
Syntax:	[ <i>label</i> ] INCFSZ f,d								
Operands:	0 ≤ f ≤ 255 d ∈ [0,1]								
Operation:	(f) + 1 → (dest) skip if result = 0								
Status Affected:	None								
Encoding:	<table border="1" style="display: inline-table;"><tr><td>0001</td><td>111d</td><td>ffff</td><td>ffff</td></tr></table>	0001	111d	ffff	ffff				
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:	<table border="1" style="display: inline-table;"><thead><tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr></thead><tbody><tr><td>Decode</td><td>Read register 'f'</td><td>Execute</td><td>Write to destination</td></tr></tbody></table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Execute	Write to destination
Q1	Q2	Q3	Q4						
Decode	Read register 'f'	Execute	Write to destination						

**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)

# PIC17C4X

## INFSNZ Increment f, skip if not 0

**Syntax:** `[label] INFSNZ 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	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'.  
 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        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:** `[label] IORLW k`

**Operands:**  $0 \leq k \leq 255$

**Operation:**  $(\text{WREG}) .\text{OR.} (k) \rightarrow (\text{WREG})$

**Status Affected:** Z

**Encoding:**

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`

**IORWF**                    **Inclusive OR WREG with f**

---

Syntax:                    [*label*] IORWF f,d

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

Operation:                (WREG) .OR. (f) → (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:                 $0 \leq k \leq 255$

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

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)  
MOVWF WREG, PCLATH  
LCALL LOW(SUBROUTINE)

Before Instruction

SUBROUTINE = 16-bit Address  
PC = ?

After Instruction

PC = Address (SUBROUTINE)

# PIC17C4X

## MOVFP Move f to p

Syntax: `[label] MOVFP f,p`

Operands:  $0 \leq f \leq 255$   
 $0 \leq p \leq 31$

Operation:  $(f) \rightarrow (p)$

Status Affected: None

Encoding: 

011p	pppp	ffff	ffff
------	------	------	------

Description: Move data from data memory location 'f' to data memory location 'p'. Location 'f' can be anywhere in the 256 word data space (00h to FFh) while 'p' can be 00h to 1Fh.

Either 'p' or 'f' can be WREG (a useful special situation).

MOVFP is particularly useful for transferring a data memory location to a peripheral register (such as the transmit buffer or an I/O port). Both 'f' and 'p' can be indirectly addressed.

Words: 1

Cycles: 1

Q Cycle Activity:

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

**Example:** `MOVFP REG1, REG2`

Before Instruction

REG1 = 0x33,  
 REG2 = 0x11

After Instruction

REG1 = 0x33,  
 REG2 = 0x33

## MOVLB Move Literal to low nibble in BSR

Syntax: `[label] MOVLB k`

Operands:  $0 \leq k \leq 15$

Operation:  $k \rightarrow (\text{BSR}\langle 3:0 \rangle)$

Status Affected: None

Encoding: 

1011	1000	uuuu	kkkk
------	------	------	------

Description: The four bit literal 'k' is loaded in the Bank Select Register (BSR). Only the low 4-bits of the Bank Select Register are affected. The upper 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 'u:k'	Execute	Write literal 'k' to BSR<3:0>

**Example:** `MOVLB 0x5`

Before Instruction

BSR register = 0x22

After Instruction

BSR register = 0x25

**Note:** For the PIC17C42, only the low four bits of the BSR register are physically implemented. The upper nibble is read as '0'.

**MOVLR**      **Move Literal to high nibble in BSR**

---

Syntax:      [ *label* ] 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:      [ *label* ] MOVLW k

Operands:     $0 \leq k \leq 255$

Operation:     $k \rightarrow (\text{WREG})$

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

# PIC17C4X

**MOVPF**      **Move p to f**

---

Syntax:        `[label] MOVPF p,f`

Operands:      $0 \leq f \leq 255$   
 $0 \leq p \leq 31$

Operation:      $(p) \rightarrow (f)$

Status Affected: Z

Encoding:     

010p	pppp	ffff	ffff
------	------	------	------

Description:    Move data from data memory location 'p' to data memory location 'f'. Location 'f' can be anywhere in the 256 byte data space (00h to FFh) while 'p' can be 00h to 1Fh.  
 Either 'p' or 'f' can be WREG (a useful special situation).  
 MOVPF is particularly useful for transferring a peripheral register (e.g. the timer or an I/O port) to a data memory location. Both 'f' and 'p' can be indirectly addressed.

Words:        1

Cycles:       1

Q Cycle Activity:

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

Example:        `MOVPF REG1, REG2`

Before Instruction

REG1        = 0x11  
 REG2        = 0x33

After Instruction

REG1        = 0x11  
 REG2        = 0x11

**MOVWF**      **Move WREG to f**

---

Syntax:        `[label] MOVWF f`

Operands:      $0 \leq f \leq 255$

Operation:      $(WREG) \rightarrow (f)$

Status Affected: None

Encoding:     

0000	0001	ffff	ffff
------	------	------	------

Description:    Move data from WREG to register 'f'. Location 'f' can be anywhere in the 256 word data space.

Words:        1

Cycles:       1

Q Cycle Activity:

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

Example:        `MOVWF REG`

Before Instruction

WREG        = 0x4F  
 REG         = 0xFF

After Instruction

WREG        = 0x4F  
 REG         = 0x4F



## MULLW Multiply Literal with WREG

**Syntax:** [ *label* ] MULLW *k*

**Operands:**  $0 \leq k \leq 255$

**Operation:**  $(k \times \text{WREG}) \rightarrow \text{PRODH}:\text{PRODL}$

**Status Affected:** None

**Encoding:**

1011	1100	kkkk	kkkk
------	------	------	------

**Description:** An unsigned multiplication is carried out between the contents of WREG and the 8-bit literal 'k'. The 16-bit result is placed in PRODH:PRODL register pair. PRODH contains the high byte. WREG is unchanged. None of the status flags are affected. Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected.

**Words:** 1

**Cycles:** 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Execute	Write registers PRODH:PRODL

**Example:** MULLW 0xC4

Before Instruction

WREG = 0xE2  
 PRODH = ?  
 PRODL = ?

After Instruction

WREG = 0xC4  
 PRODH = 0xAD  
 PRODL = 0x08

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

## MULWF Multiply WREG with f

**Syntax:** [ *label* ] MULWF *f*

**Operands:**  $0 \leq f \leq 255$

**Operation:**  $(\text{WREG} \times f) \rightarrow \text{PRODH}:\text{PRODL}$

**Status Affected:** None

**Encoding:**

0011	0100	ffff	ffff
------	------	------	------

**Description:** An unsigned multiplication is carried out between the contents of WREG and the register file location 'f'. The 16-bit result is stored in the PRODH:PRODL register pair. PRODH contains the high byte. Both WREG and 'f' are unchanged. None of the status flags are affected. Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected.

**Words:** 1

**Cycles:** 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Execute	Write registers PRODH:PRODL

**Example:** MULWF REG

Before Instruction

WREG = 0xC4  
 REG = 0xB5  
 PRODH = ?  
 PRODL = ?

After Instruction

WREG = 0xC4  
 REG = 0xB5  
 PRODH = 0x8A  
 PRODL = 0x94

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

# PIC17C4X

## NEGW

### Negate W

Syntax: `[label] NEGW f,s`

Operands:  $0 \leq F \leq 255$   
 $s \in [0,1]$

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

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.

## RETFIE Return from Interrupt

Syntax: [ *label* ] RETFIE

Operands: None

Operation: TOS → (PC);  
0 → GLINTD;  
PCLATH is unchanged.

Status Affected: GLINTD

Encoding: 

0000	0000	0000	0101
------	------	------	------

Description: Return from Interrupt. Stack is POP'ed and Top of Stack (TOS) is loaded in the PC. Interrupts are enabled by clearing the GLINTD bit. GLINTD is the global interrupt disable bit (CPUSTA<4>).

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register TOSTA	Execute	NOP
Forced NOP	NOP	Execute	NOP

**Example:** RETFIE

After Interrupt  
PC = TOS  
GLINTD = 0

## RETLW Return Literal to WREG

Syntax: [ *label* ] RETLW k

Operands:  $0 \leq k \leq 255$

Operation: k → (WREG); TOS → (PC);  
PCLATH is unchanged

Status Affected: None

Encoding: 

1011	0110	kkkk	kkkk
------	------	------	------

Description: WREG is loaded with the eight bit literal 'k'. The program counter is loaded from the top of the stack (the return address). The high address latch (PCLATH) remains unchanged.

Words: 1

Cycles: 2

Q Cycle Activity:

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

**Example:**

```
CALL TABLE ; WREG contains table
              ; offset value
              ; WREG now has
              ; table value
:
TABLE
ADDWF PC ; WREG = offset
RETLW k0 ; Begin table
RETLW k1 ;
:
:
RETLW kn ; End of table
```

Before Instruction  
WREG = 0x07

After Instruction  
WREG = value of k7

# PIC17C4X

## RETURN Return from Subroutine

Syntax: [ *label* ] RETURN

Operands: None

Operation: TOS → PC;

Status Affected: None

Encoding: 

0000	0000	0000	0010
------	------	------	------

Description: Return from subroutine. The stack is popped and the top of the stack (TOS) is loaded into the program counter.

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register PCL*	Execute	NOP
Forced NOP	NOP	Execute	NOP

\* Remember reading PCL causes PCLATH to be updated. This will be the high address of where the RETURN instruction is located.

**Example:** RETURN

After Interrupt  
PC = TOS

## RLCF Rotate Left f through Carry

Syntax: [ *label* ] RLCF f,d

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

Operation:  $f\langle n \rangle \rightarrow d\langle n+1 \rangle$ ;  
 $f\langle 7 \rangle \rightarrow C$ ;  
 $C \rightarrow d\langle 0 \rangle$

Status Affected: C

Encoding: 

0001	101d	ffff	ffff
------	------	------	------

Description: The contents of register 'f' are rotated one bit to the left through the Carry Flag. If 'd' is 0 the result is placed 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:** RLCF REG, 0

Before Instruction

REG = 1110 0110  
C = 0

After Instruction

REG = 1110 0110  
WREG = 1100 1100  
C = 1

## RLNCF Rotate Left f (no carry)

Syntax: [label] RLNCF f,d

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

Operation:  $f\langle n \rangle \rightarrow d\langle n+1 \rangle$ ;  
 $f\langle 7 \rangle \rightarrow d\langle 0 \rangle$

Status Affected: None

Encoding: 

0010	001d	ffff	ffff
------	------	------	------

Description: The contents of register 'f' are rotated one bit to the left. If 'd' is 0 the result is placed 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:** RLNCF REG, 1

Before Instruction

C = 0  
 REG = 1110 1011

After Instruction

C =  
 REG = 1101 0111

## RRCF Rotate Right f through Carry

Syntax: [label] RRCF f,d

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

Operation:  $f\langle n \rangle \rightarrow d\langle n-1 \rangle$ ;  
 $f\langle 0 \rangle \rightarrow C$ ;  
 $C \rightarrow d\langle 7 \rangle$

Status Affected: C

Encoding: 

0001	100d	ffff	ffff
------	------	------	------

Description: The contents of register 'f' are rotated one bit to the right through the Carry Flag. 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:** RRCF REG1, 0

Before Instruction

REG1 = 1110 0110  
 C = 0

After Instruction

REG1 = 1110 0110  
 WREG = 0111 0011  
 C = 0

# PIC17C4X

## RRNCF Rotate Right f (no carry)

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

Operation:  $f \langle n \rangle \rightarrow d \langle n-1 \rangle$ ;  
 $f \langle 0 \rangle \rightarrow d \langle 7 \rangle$

Status Affected: None

Encoding: 

0010	000d	ffff	ffff
------	------	------	------

Description: The contents of register 'f' are rotated one bit to the right. 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 1:** RRNCF REG, 1

Before Instruction  
 WREG = ?  
 REG = 1101 0111

After Instruction  
 WREG = 0  
 REG = 1110 1011

**Example 2:** RRNCF REG, 0

Before Instruction  
 WREG = ?  
 REG = 1101 0111

After Instruction  
 WREG = 1110 1011  
 REG = 1101 0111

## SETF Set f

Syntax: [label] SETF f,s  
 Operands:  $0 \leq f \leq 255$   
 $s \in [0,1]$

Operation: FFh  $\rightarrow$  f;  
 FFh  $\rightarrow$  d

Status Affected: None

Encoding: 

0010	101s	ffff	ffff
------	------	------	------

Description: If 's' is 0, both the data memory location 'f' and WREG are set to FFh. If 's' is 1 only the data memory location 'f' is set to FFh.

Words: 1  
 Cycles: 1

Q Cycle Activity:

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

**Example 1:** SETF REG, 0

Before Instruction  
 REG = 0xDA  
 WREG = 0x05

After Instruction  
 REG = 0xFF  
 WREG = 0xFF

**Example 2:** SETF REG, 1

Before Instruction  
 REG = 0xDA  
 WREG = 0x05

After Instruction  
 REG = 0xFF  
 WREG = 0x05

## SLEEP Enter SLEEP mode

Syntax: [ *label* ] SLEEP

Operands: None

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

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

Encoding: 

0000	0000	0000	0011
------	------	------	------

Description: The power down status bit ( $\overline{PD}$ ) is cleared. The time-out status bit ( $\overline{TO}$ ) is set. Watchdog Timer and its prescaler are cleared.  
The processor is put into SLEEP mode with the oscillator stopped.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register PCLATH	Execute	NOP

**Example:** SLEEP

Before Instruction

$\overline{TO}$  = ?

$\overline{PD}$  = ?

After Instruction

$\overline{TO}$  = 1 †

$\overline{PD}$  = 0

† If WDT causes wake-up, this bit is cleared

## SUBLW Subtract WREG from Literal

Syntax: [ *label* ] SUBLW *k*

Operands:  $0 \leq k \leq 255$

Operation:  $k - (WREG) \rightarrow (WREG)$

Status Affected: OV, C, DC, Z

Encoding: 

1011	0010	kkkk	kkkk
------	------	------	------

Description: WREG is subtracted from 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 1:** SUBLW 0x02

Before Instruction

WREG = 1

C = ?

After Instruction

WREG = 1

C = 1 ; result is positive

Z = 0

**Example 2:**

Before Instruction

WREG = 2

C = ?

After Instruction

WREG = 0

C = 1 ; result is zero

Z = 1

**Example 3:**

Before Instruction

WREG = 3

C = ?

After Instruction

WREG = FF ; (2's complement)

C = 0 ; result is negative

Z = 1

# 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:  $(f) - (W) - \bar{C} \rightarrow (\text{dest})$   
 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

**Example 2:** 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

**Example 3:** 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



**SWAPF**      **Swap f**

---

Syntax:      [ *label* ] SWAPF f,d

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

Operation:    f<3:0> → dest<7:4>;  
 f<7:4> → 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

**Example:**      SWAPF REG, 0

Before Instruction  
 REG = 0x53

After Instruction  
 REG = 0x35

**TABLRD**      **Table Read**

---

Syntax:      [ *label* ] TABLRD t,i,f

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

Operation:    If t = 1,  
                   TBLATH → f;  
 If t = 0,  
                   TBLATL → f;  
                   Prog Mem (TBLPTR) → TBLAT;  
 If i = 1,  
                   TBLPTR + 1 → 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'

# 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  $t = 0$ ,  
 $f \rightarrow$  TBLATL;  
 If  $t = 1$ ,  
 $f \rightarrow$  TBLATH;  
 TBLAT  $\rightarrow$  Prog Mem (TBLPTR);  
 If  $i = 1$ ,  
 TBLPTR + 1  $\rightarrow$  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  $\overline{\text{MCLR}}/\text{VPP}$  pin must be at the programming voltage for successful programming of internal memory.  
 If  $\overline{\text{MCLR}}/\text{VPP} = \text{VDD}$  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

## TABLWT Table Write

**Example 1:** TABLWT 0, 1, REG

Before Instruction

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

After Instruction (table write completion)

```
REG          = 0x53
TBLATH       = 0x53
TBLATL       = 0x55
TBLPTR       = 0xA357
MEMORY(TBLPTR - 1) = 0x5355
```

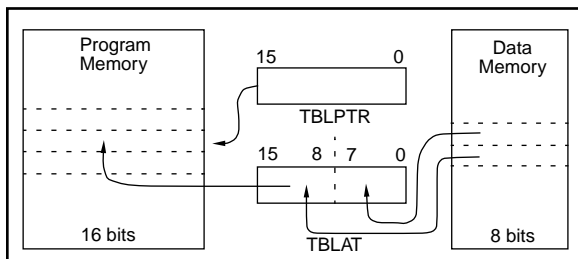
**Example 2:** TABLWT 1, 0, REG

Before Instruction

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

After Instruction (table write completion)

```
REG          = 0x53
TBLATH       = 0xAA
TBLATL       = 0x53
TBLPTR       = 0xA356
MEMORY(TBLPTR) = 0xAA53
```



## TLRD Table Latch Read

**Syntax:** [label] TLRD t,f

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

**Operation:** If  $t = 0$ ,  
 TBLATL  $\rightarrow$  f;  
 If  $t = 1$ ,  
 TBLATH  $\rightarrow$  f

**Status Affected:** None

**Encoding:**

1010	00tx	ffff	ffff
------	------	------	------

**Description:** Read data from 16-bit table latch (TBLAT) into file register 'f'. Table Latch is unaffected.

If  $t = 1$ ; high byte is read

If  $t = 0$ ; low byte is read

This instruction is used in conjunction with TABLWD to transfer data from program memory to data memory.

**Words:** 1

**Cycles:** 1

**Q Cycle Activity:**

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

**Example:** TLRD t, RAM

Before Instruction

```
t          = 0
RAM        = ?
TBLAT      = 0x00AF (TBLATH = 0x00)
              (TBLATL = 0xAF)
```

After Instruction

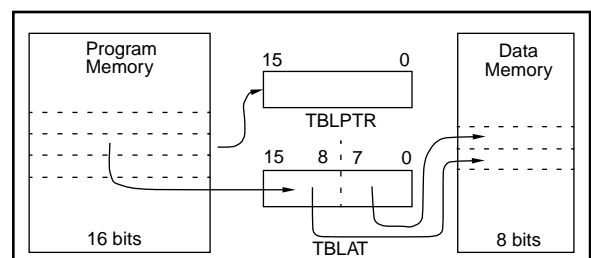
```
RAM        = 0xAF
TBLAT      = 0x00AF (TBLATH = 0x00)
              (TBLATL = 0xAF)
```

Before Instruction

```
t          = 1
RAM        = ?
TBLAT      = 0x00AF (TBLATH = 0x00)
              (TBLATL = 0xAF)
```

After Instruction

```
RAM        = 0x00
TBLAT      = 0x00AF (TBLATH = 0x00)
              (TBLATL = 0xAF)
```



# PIC17C4X

## TLWT **Table Latch Write**

**Syntax:** [ *label* ] TLWT t,f

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

**Operation:** If  $t = 0$ ,  
 $f \rightarrow \text{TBLATL}$ ;  
 If  $t = 1$ ,  
 $f \rightarrow \text{TBLATH}$

**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:**  $0 \leq f \leq 255$

**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  $\neq$  0x00,  
 PC = Address ( NZERO )

**XORLW**                    **Exclusive OR Literal with WREG**

---

Syntax:                    [ *label* ] XORLW *k*

Operands:                 $0 \leq k \leq 255$

Operation:                (WREG) .XOR. *k* → (WREG)

Status Affected:        Z

Encoding:                

1011	0100	kkkk	kkkk
------	------	------	------

Description:             The contents of WREG are XOR'ed with the 8-bit literal '*k*'. The result is placed in WREG.

Words:                    1

Cycles:                   1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal ' <i>k</i> '	Execute	Write to WREG

**Example:**                    XORLW    0xAF

Before Instruction  
WREG = 0xB5

After Instruction  
WREG = 0x1A

**XORWF**                    **Exclusive OR WREG with f**

---

Syntax:                    [ *label* ] XORWF *f,d*

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

Operation:                (WREG) .XOR. (*f*) → (*dest*)

Status Affected:        Z

Encoding:                

0000	110d	ffff	ffff
------	------	------	------

Description:             Exclusive OR the contents of WREG with register '*f*'. If '*d*' is 0 the result is stored in WREG. If '*d*' is 1 the result is stored back in the register '*f*'.

Words:                    1

Cycles:                   1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register ' <i>f</i> '	Execute	Write to destination

**Example:**                    XORWF    REG, 1

Before Instruction  
REG = 0xAF  
WREG = 0xB5

After Instruction  
REG = 0x1A  
WREG = 0xB5

# PIC17C4X

---

NOTES:

## 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 VDD and VPP supplies which allows it to verify programmed memory at VDD min and VDD 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.

## 16.6 PICDEM-1 Low-Cost PIC16/17 Demonstration Board

The PICDEM-1 is a simple board which demonstrates the capabilities of several of Microchip's microcontrollers. The microcontrollers supported are: PIC16C5X (PIC16C54 to PIC16C58A), PIC16C61, PIC16C62X, PIC16C71, PIC16C8X, PIC17C42, PIC17C43 and PIC17C44. All necessary hardware and software is included to run basic demo programs. The users can program the sample microcontrollers provided with the PICDEM-1 board, on a PRO MATE II or PICSTART-16B programmer, and easily test firmware. The user can also connect the PICDEM-1 board to the PICMASTER emulator and download the firmware to the emulator for testing. Additional prototype area is available for the user to build some additional hardware and connect it to the microcontroller socket(s). Some of the features include an RS-232 interface, a potentiometer for simulated analog input, push-button switches and eight LEDs connected to PORTB.

## 16.7 PICDEM-2 Low-Cost PIC16CXX Demonstration Board

The PICDEM-2 is a simple demonstration board that supports the PIC16C62, PIC16C64, PIC16C65, PIC16C73 and PIC16C74 microcontrollers. All the necessary hardware and software is included to run the basic demonstration programs. The user can program the sample microcontrollers provided with the PICDEM-2 board, on a PRO MATE II programmer or PICSTART-16C, and easily test firmware. The PICMASTER emulator may also be used with the PICDEM-2 board to test firmware. Additional prototype area has been provided to the user for adding additional hardware and connecting it to the microcontroller socket(s). Some of the features include a RS-232 interface, push-button switches, a potentiometer for simulated analog input, a Serial EEPROM to demonstrate usage of the I<sup>2</sup>C bus and separate headers for connection to an LCD module and a keypad.

## 16.8 PICDEM-3 Low-Cost PIC16CXXX Demonstration Board

The PICDEM-3 is a simple demonstration board that supports the PIC16C923 and PIC16C924 in the PLCC package. It will also support future 44-pin PLCC microcontrollers with a LCD Module. All the necessary hardware and software is included to run the basic demonstration programs. The user can program the sample microcontrollers provided with the PICDEM-3 board, on a PRO MATE II programmer or PICSTART Plus with an adapter socket, and easily test firmware. The PICMASTER emulator may also be used with the PICDEM-3 board to test firmware. Additional prototype area has been provided to the user for adding hardware and connecting it to the microcontroller socket(s). Some of the features

include an RS-232 interface, push-button switches, a potentiometer for simulated analog input, a thermistor and separate headers for connection to an external LCD module and a keypad. Also provided on the PICDEM-3 board is an LCD panel, with 4 commons and 12 segments, that is capable of displaying time, temperature and day of the week. The PICDEM-3 provides an additional RS-232 interface and Windows 3.1 software for showing the demultiplexed LCD signals on a PC. A simple serial interface allows the user to construct a hardware demultiplexer for the LCD signals. PICDEM-3 will be available in the 3rd quarter of 1996.

## 16.9 MPLAB Integrated Development Environment Software

The MPLAB IDE Software brings an ease of software development previously unseen in the 8-bit microcontroller market. MPLAB is a windows based application which contains:

- A full featured editor
- Three operating modes
  - editor
  - emulator
  - simulator
- A project manager
- Customizable tool bar and key mapping
- A status bar with project information
- Extensive on-line help

MPLAB allows you to:

- Edit your source files (either assembly or 'C')
- One touch assemble (or compile) and download to PIC16/17 tools (automatically updates all project information)
- Debug using:
  - source files
  - absolute listing file
- Transfer data dynamically via DDE (soon to be replaced by OLE)
- Run up to four emulators on the same PC

The ability to use MPLAB with Microchip's simulator allows a consistent platform and the ability to easily switch from the low cost simulator to the full featured emulator with minimal retraining due to development tools.

## 16.10 Assembler (MPASM)

The MPASM Universal Macro Assembler is a PC-hosted symbolic assembler. It supports all microcontroller series including the PIC12C5XX, PIC14000, PIC16C5X, PIC16CXXX, and PIC17CXX families.

MPASM offers full featured Macro capabilities, conditional assembly, and several source and listing formats. It generates various object code formats to support Microchip's development tools as well as third party programmers.



MPASM allow full symbolic debugging from the Microchip Universal Emulator System (PICMASTER).

MPASM has the following features to assist in developing software for specific use applications.

- Provides translation of Assembler source code to object code for all Microchip microcontrollers.
- Macro assembly capability.
- Produces all the files (Object, Listing, Symbol, and special) required for symbolic debug with Microchip's emulator systems.
- Supports Hex (default), Decimal and Octal source and listing formats.

MPASM provides a rich directive language to support programming of the PIC16/17. Directives are helpful in making the development of your assemble source code shorter and more maintainable.

## **16.11 Software Simulator (MPLAB-SIM)**

The MPLAB-SIM Software Simulator allows code development in a PC host environment. It allows the user to simulate the PIC16/17 series microcontrollers on an instruction level. On any given instruction, the user may examine or modify any of the data areas or provide external stimulus to any of the pins. The input/output radix can be set by the user and the execution can be performed in; single step, execute until break, or in a trace mode.

MPLAB-SIM fully supports symbolic debugging using MPLAB-C and MPASM. The Software Simulator offers the low cost flexibility to develop and debug code outside of the laboratory environment making it an excellent multi-project software development tool.

## **16.12 C Compiler (MPLAB-C)**

The MPLAB-C Code Development System is a complete 'C' compiler and integrated development environment for Microchip's PIC16/17 family of microcontrollers. The compiler provides powerful integration capabilities and ease of use not found with other compilers.

For easier source level debugging, the compiler provides symbol information that is compatible with the MPLAB IDE memory display (PICMASTER emulator software versions 1.13 and later).

## **16.13 Fuzzy Logic Development System (fuzzyTECH-MP)**

*fuzzyTECH-MP* fuzzy logic development tool is available in two versions - a low cost introductory version, MP Explorer, for designers to gain a comprehensive working knowledge of fuzzy logic system design; and a full-featured version, *fuzzyTECH-MP*, edition for implementing more complex systems.

Both versions include Microchip's *fuzzyLAB*<sup>™</sup> demonstration board for hands-on experience with fuzzy logic systems implementation.

## **16.14 MP-DriveWay<sup>™</sup> – Application Code Generator**

MP-DriveWay is an easy-to-use Windows-based Application Code Generator. With MP-DriveWay you can visually configure all the peripherals in a PIC16/17 device and, with a click of the mouse, generate all the initialization and many functional code modules in C language. The output is fully compatible with Microchip's MPLAB-C C compiler. The code produced is highly modular and allows easy integration of your own code. MP-DriveWay is intelligent enough to maintain your code through subsequent code generation.

## **16.15 SEEVAL<sup>®</sup> Evaluation and Programming System**

The SEEVAL SEEPROM Designer's Kit supports all Microchip 2-wire and 3-wire Serial EEPROMs. The kit includes everything necessary to read, write, erase or program special features of any Microchip SEEPROM product including Smart Serial<sup>™</sup> and secure serials. The Total Endurance<sup>™</sup> Disk is included to aid in trade-off analysis and reliability calculations. The total kit can significantly reduce time-to-market and result in an optimized system.

## **16.16 TrueGauge<sup>®</sup> Intelligent Battery Management**

The TrueGauge development tool supports system development with the MTA11200B TrueGauge Intelligent Battery Management IC. System design verification can be accomplished before hardware prototypes are built. User interface is graphically-oriented and measured data can be saved in a file for exporting to Microsoft Excel.

## **16.17 KEELOQ<sup>®</sup> Evaluation and Programming Tools**

KEELOQ evaluation and programming tools support Microchips HCS Secure Data Products. The HCS evaluation kit includes an LCD display to show changing codes, a decoder to decode transmissions, and a programming interface to program test transmitters.

**TABLE 16-1: DEVELOPMENT TOOLS FROM MICROCHIP**

Product	** MPLAB™ Integrated Development Environment	MPLAB™ C Compiler	MP-DriveWay Applications Code Generator	fuzzyTECH®-MP Explorer/Edition Fuzzy Logic Dev. Tool	*** PICMASTER®/PICMASTER-CE In-Circuit Emulator	ICEPIC Low-Cost In-Circuit Emulator	****PRO MATE™ II Universal Microchip Programmer	PICSTART® Lite Ultra Low-Cost Dev. Kit	PICSTART® Plus Low-Cost Universal Dev. Kit
PIC12C508, 509	SW007002	SW006005	—	—	EM167015/ EM167101	—	DV007003	—	DV003001
PIC14000	SW007002	SW006005	—	—	EM147001/ EM147101	—	DV007003	—	DV003001
PIC16C52, 54, 54A, 55, 56, 57, 58A	SW007002	SW006005	SW006006	DV005001/ DV005002	EM167015/ EM167101	EM167201	DV007003	DV162003	DV003001
PIC16C55A, 556, 558	SW007002	SW006005	—	DV005001/ DV005002	EM167033/ EM167113	—	DV007003	—	DV003001
PIC16C61	SW007002	SW006005	SW006006	DV005001/ DV005002	EM167021/ N/A	EM167205	DV007003	DV162003	DV003001
PIC16C62, 62A, 64, 64A	SW007002	SW006005	SW006006	DV005001/ DV005002	EM167025/ EM167103	EM167203	DV007003	DV162002	DV003001
PIC16C620, 621, 622	SW007002	SW006005	SW006006	DV005001/ DV005002	EM167023/ EM167109	EM167202	DV007003	DV162003	DV003001
PIC16C63, 65, 65A, 73, 73A, 74, 74A	SW007002	SW006005	SW006006	DV005001/ DV005002	EM167025/ EM167103	EM167204	DV007003	DV162002	DV003001
PIC16C642, 662*	SW007002	SW006005	—	—	EM167035/ EM167105	—	DV007003	DV162002	DV003001
PIC16C71	SW007002	SW006005	SW006006	DV005001/ DV005002	EM167027/ EM167105	EM167205	DV007003	DV162003	DV003001
PIC16C710, 711	SW007002	SW006005	SW006006	DV005001/ DV005002	EM167027/ EM167105	—	DV007003	DV162003	DV003001
PIC16C72	SW007002	SW006005	SW006006	—	EM167025/ EM167103	—	DV007003	DV162002	DV003001
PIC16F83	SW007002	SW006005	SW006006	DV005001/ DV005002	EM167029/ EM167107	—	DV007003	DV162003	DV003001
PIC16C84	SW007002	SW006005	SW006006	DV005001/ DV005002	EM167029/ EM167107	EM167206	DV007003	DV162003	DV003001
PIC16F84	SW007002	SW006005	SW006006	DV005001/ DV005002	EM167029/ EM167107	—	DV007003	DV162003	DV003001
PIC16C923, 924*	SW007002	SW006005	SW006006	DV005001/ DV005002	EM167031/ EM167111	—	DV007003	—	DV003001
PIC17C42, 42A, 43, 44	SW007002	SW006005	SW006006	DV005001/ DV005002	EM177007/ EM177107	—	DV007003	—	DV003001

\*Contact Microchip Technology for availability date  
 \*\*MPLAB Integrated Development Environment includes MPLAB-SIM Simulator and MPA/SM Assembler  
 \*\*\*All PICMASTER and PICMASTER-CE ordering part numbers above include PRO MATE II programmer  
 \*\*\*\*PRO MATE socket modules are ordered separately. See development systems ordering guide for specific ordering part numbers

Product	TRUEGAUGE® Development Kit	SEEVAL® Designers Kit	Hopping Code Security Programmer Kit	Hopping Code Security Eval/Demo Kit
All 2 wire and 3 wire Serial EEPROM's	N/A	DV243001	N/A	N/A
MTA11200B	DV114001	N/A	N/A	N/A
HCS200, 300, 301 *	N/A	N/A	PG306001	DM303001

## 17.0 PIC17C42 ELECTRICAL CHARACTERISTICS

### Absolute Maximum Ratings †

Ambient temperature under bias.....	-55 to +125°C
Storage temperature .....	-65°C to +150°C
Voltage on VDD with respect to VSS .....	0 to +7.5V
Voltage on $\overline{\text{MCLR}}$ with respect to VSS (Note 2) .....	-0.6V to +14V
Voltage on RA2 and RA3 with respect to VSS.....	-0.6V to +12V
Voltage on all other pins with respect to VSS .....	-0.6V to VDD + 0.6V
Total power dissipation (Note 1).....	1.0W
Maximum current out of VSS pin(s) - Total .....	250 mA
Maximum current into VDD pin(s) - Total .....	200 mA
Input clamp current, I <sub>IK</sub> (V <sub>I</sub> < 0 or V <sub>I</sub> > VDD) .....	±20 mA
Output clamp current, I <sub>OK</sub> (V <sub>O</sub> < 0 or V <sub>O</sub> > VDD).....	±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:  $P_{dis} = V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD} - V_{OH}) \times I_{OH}\} + \sum (V_{OL} \times I_{OL})$

**Note 2:** Voltage spikes below VSS at the  $\overline{\text{MCLR}}$  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  $\overline{\text{MCLR}}$  pin rather than pulling this pin directly to VSS.

† 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

Applicable Devices 42 R42 42A 43 R43 44

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

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

**17.1 DC CHARACTERISTICS:**      **PIC17C42-16 (Commercial, Industrial)**  
**PIC17C42-25 (Commercial, Industrial)**

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					
Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
D001	VDD	Supply Voltage	4.5	–	5.5	V	
D002	VDR	RAM Data Retention Voltage (Note 1)	1.5 *	–	–	V	Device in SLEEP mode
D003	VPOR	VDD start voltage to ensure internal Power-on Reset signal	–	VSS	–	V	See section on Power-on Reset for details
D004	SVDD	VDD rise rate to ensure internal Power-on Reset signal	0.060*	–	–	mV/ms	See section on Power-on Reset for details
D010	IDD	Supply Current (Note 2)	–	3	6	mA	FOSC = 4 MHz (Note 4)
D011			–	6	12 *	mA	FOSC = 8 MHz
D012			–	11	24 *	mA	FOSC = 16 MHz
D013			–	19	38	mA	FOSC = 25 MHz
D014			–	95	150	µA	FOSC = 32 kHz WDT enabled (EC osc configuration)
D020	IPD	Power-down Current (Note 3)	–	10	40	µA	VDD = 5.5V, WDT enabled
D021			–	< 1	5	µA	VDD = 5.5V, WDT disabled

\* 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.

Note 1: This is the limit to which VDD can be lowered in SLEEP mode without losing RAM data.

2: The supply current is mainly a function of the operating voltage and frequency. Other factors such as I/O pin loading and switching rate, oscillator type, internal code execution pattern, and temperature also have an impact on the current consumption.

The test conditions for all IDD measurements in active operation mode are:

OSC1 = external square wave, from rail to rail; all I/O pins tristated, pulled to VDD or VSS, T0CKI = VDD, MCLR = VDD; WDT enabled/disabled as specified.

Current consumed from the oscillator and I/O's driving external capacitive or resistive loads need to be considered.

For the RC oscillator, the current through the external pull-up resistor (R) can be estimated as:  $V_{DD} / (2 \cdot R)$ .

For capacitive loads, The current can be estimated (for an individual I/O pin) as  $(C_L \cdot V_{DD}) \cdot f$

CL = Total capacitive load on the I/O pin; f = average frequency on the I/O pin switches.

The capacitive currents are most significant when the device is configured for external execution (includes extended microcontroller mode).

3: The power-down current in SLEEP mode does not depend on the oscillator type. Power-down current is measured with the part in SLEEP mode, all I/O pins in hi-impedance state and tied to VDD or VSS.

4: For RC osc configuration, current through Rext is not included. The current through the resistor can be estimated by the formula  $I_R = V_{DD}/2R_{ext}$  (mA) with Rext in kOhm.

# PIC17C4X

Applicable Devices 42 R42 42A 43 R43 44

## 17.2 DC CHARACTERISTICS: PIC17C42-16 (Commercial, Industrial) PIC17C42-25 (Commercial, Industrial)

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 17.1							
Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
<b>DC CHARACTERISTICS</b>							
D030	VIL	<b>Input Low Voltage</b> I/O ports with TTL buffer	VSS	–	0.8	V	Note1
D031		with Schmitt Trigger buffer	VSS	–	0.2VDD	V	
D032		MCLR, OSC1 (in EC and RC mode)	VSS	–	0.2VDD	V	
D033		OSC1 (in XT, and LF mode)	–	0.5VDD	–	V	
D040	VIH	<b>Input High Voltage</b> I/O ports with TTL buffer	2.0	–	VDD	V	Note1
D041		with Schmitt Trigger buffer	0.8VDD	–	VDD	V	
D042		MCLR	0.8VDD	–	VDD	V	
D043		OSC1 (XT, and LF mode)	–	0.5VDD	–	V	
D050	VHYS	Hysteresis of Schmitt Trigger inputs	0.15VDD*	–	–	V	
D060	IIL	<b>Input Leakage Current</b> (Notes 2, 3) I/O ports (except RA2, RA3)	–	–	±1	µA	VSS ≤ VPIN ≤ VDD, I/O Pin at hi-impedance PORTB weak pull-ups dis- abled
D061		MCLR	–	–	±2	µA	VPIN = VSS or VPIN = VDD
D062		RA2, RA3	–	–	±2	µA	VSS ≤ VRA2, VRA3 ≤ 12V
D063		OSC1, TEST	–	–	±1	µA	VSS ≤ VPIN ≤ VDD
D064		MCLR	–	–	10	µA	VMCLR = VPP = 12V (when not programming)
D070	IPURB	PORTB weak pull-up current	60	200	400	µA	VPIN = VSS, RBPU = 0

\* 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.

†† Design guidance to attain the AC timing specifications. These loads are not tested.

- Note 1: In RC oscillator configuration, the OSC1 pin is a Schmitt Trigger input. It is not recommended that the PIC17CXX devices be driven with external clock in RC mode.
- The leakage current on the 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.
  - Negative current is defined as coming out of the pin.
  - 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).
  - The MCLR/Vpp pin may be kept in this range at times other than programming, but this is not recommended.
  - For TTL buffers, the better of the two specifications may be used.

# PIC17C4X

Applicable Devices 42 R42 42A 43 R43 44

Standard Operating Conditions (unless otherwise stated)							
DC CHARACTERISTICS							
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 17.1							
Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
D080	VOL	<b>Output Low Voltage</b>					
D081		I/O ports (except RA2 and RA3) with TTL buffer	–	–	0.1VDD	V	IOL = 4 mA
D082		RA2 and RA3 OSC2/CLKOUT (RC and EC osc modes)	–	–	3.0	V	IOL = 6 mA, VDD = 4.5V
D083			–	–	0.4	V	Note 6 IOL = 60.0 mA, VDD = 5.5V IOL = 2 mA, VDD = 4.5V
D090	VOH	<b>Output High Voltage (Note 3)</b>					
D091		I/O ports (except RA2 and RA3) with TTL buffer	0.9VDD	–	–	V	IOH = -2 mA
D092		RA2 and RA3	–	–	12	V	IOH = -6.0 mA, VDD = 4.5V
D093			OSC2/CLKOUT (RC and EC osc modes)	2.4	–	–	V
D100	Cosc2	<b>Capacitive Loading Specs on Output Pins</b> OSC2 pin	–	–	25 ††	pF	In EC or RC osc modes when OSC2 pin is outputting CLKOUT. External clock is used to drive OSC1.
D101	CIO	All I/O pins and OSC2 (in RC mode)	–	–	50 ††	pF	
D102	CAD	System Interface Bus (PORTC, PORTD and PORTE)	–	–	100 ††	pF	

\* 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.

†† Design guidance to attain the AC timing specifications. These loads are not tested.

Note 1: In RC oscillator configuration, the OSC1 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 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 MCLR/Vpp pin may be kept in this range at times other than programming, but this is not recommended.

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

# PIC17C4X

Applicable Devices 42 R42 42A 43 R43 44

DC CHARACTERISTICS		Standard Operating Conditions (unless otherwise stated)					
		Operating temperature -40°C ≤ T <sub>A</sub> ≤ +40°C					
		Operating voltage V <sub>DD</sub> range as described in Section 17.1					
Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
		<b>Internal Program Memory Programming Specs (Note 4)</b>					
D110	V <sub>PP</sub>	Voltage on $\overline{\text{MCLR}}/\text{VPP}$ pin	12.75	–	13.25	V	Note 5
D111	V <sub>DDP</sub>	Supply voltage during programming	4.75	5.0	5.25	V	
D112	I <sub>PP</sub>	Current into $\overline{\text{MCLR}}/\text{VPP}$ pin	–	25 ‡	50 ‡	mA	
D113	I <sub>DDP</sub>	Supply current during programming	–	–	30 ‡	mA	
D114	T <sub>PROG</sub>	Programming pulse width	10	100	1000	μs	Terminated via internal/external interrupt or a reset

\* 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.
- The leakage current on the  $\overline{\text{MCLR}}$  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.
  - Negative current is defined as coming out of the pin.
  - 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).
  - The  $\overline{\text{MCLR}}/\text{VPP}$  pin may be kept in this range at times other than programming, but is not recommended.
  - For TTL buffers, the better of the two specifications may be used.

**Note:** When using the Table Write for internal programming, the device temperature must be less than 40°C.



## 17.3 Timing Parameter Symbology

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

1. TppS2ppS
2. TppS

<b>T</b>			
F	Frequency	T	Time

Lowercase symbols (pp) and their meanings:

<b>pp</b>			
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:

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

# PIC17C4X

Applicable Devices 42 R42 42A 43 R43 44

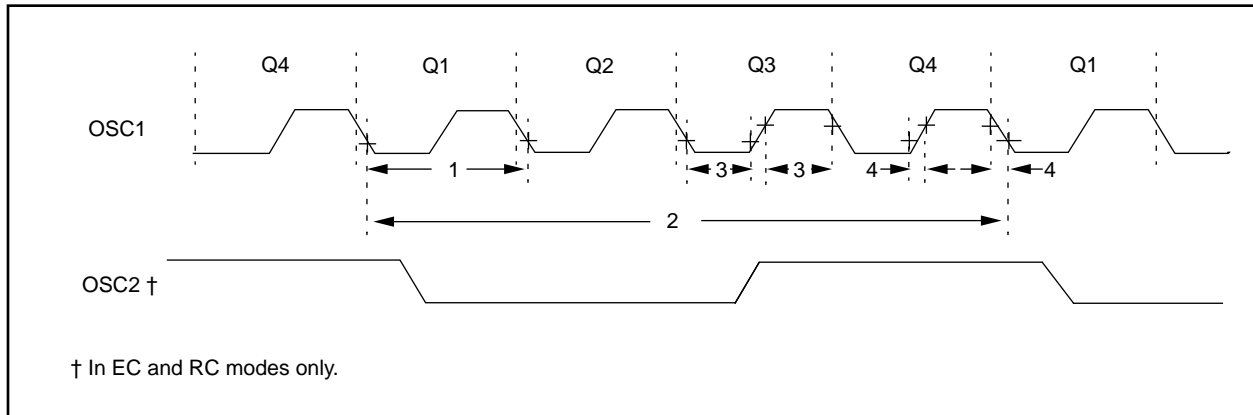
**FIGURE 17-1: PARAMETER MEASUREMENT INFORMATION**

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



## 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	<b>External CLKIN Frequency (Note 1)</b>	DC	—	16	MHz	EC osc mode - PIC17C42-16
			DC	—	25	MHz	- PIC17C42-25
		<b>Oscillator Frequency (Note 1)</b>	DC	—	4	MHz	RC osc mode
			1	—	16	MHz	XT osc mode - PIC17C42-16
			1	—	25	MHz	- PIC17C42-25
			DC	—	2	MHz	LF osc mode
1	Tosc	<b>External CLKIN Period (Note 1)</b>	62.5	—	—	ns	EC osc mode - PIC17C42-16
			40	—	—	ns	- PIC17C42-25
		<b>Oscillator Period (Note 1)</b>	250	—	—	ns	RC osc mode
			62.5	—	1,000	ns	XT osc mode - PIC17C42-16
			40	—	1,000	ns	- PIC17C42-25
			500	—	—	ns	LF osc mode
2	TCY	<b>Instruction Cycle Time (Note 1)</b>	160	4/Fosc	DC	ns	
3	TosL, TosH	<b>Clock in (OSC1) High or Low Time</b>	10 ‡	—	—	ns	EC oscillator
4	TosR, TosF	<b>Clock in (OSC1) Rise or Fall Time</b>	—	—	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.

# 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	TckH2ioL	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 17-4: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER TIMING**



**TABLE 17-4: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER REQUIREMENTS**

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
30	Tmcl	MCLR Pulse Width (low)	100 *	—	—	ns	
31	Twdt	Watchdog Timer Time-out Period (Prescale = 1)	5 *	12	25 *	ms	
32	Tost	Oscillation Start-up Timer Period		1024 T <sub>osc</sub> §		ms	T <sub>osc</sub> = OSC1 period
33	Tpwrt	Power-up Timer Period	40 *	96	200 *	ms	
35	Tmcl2adl	MCLR to System Interface bus (AD15:AD0) invalid	—	—	100 *	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.

§ This specification ensured by design.

# PIC17C4X

Applicable Devices 42 R42 42A 43 R43 44

**FIGURE 17-5: TIMER0 CLOCK TIMINGS**



**TABLE 17-5: TIMER0 CLOCK REQUIREMENTS**

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
40	Tt0H	T0CKI High Pulse Width	No Prescaler	$0.5T_{CY} + 20$ §	—	—	ns
			With Prescaler	$10^*$	—	—	ns
41	Tt0L	T0CKI Low Pulse Width	No Prescaler	$0.5T_{CY} + 20$ §	—	—	ns
			With Prescaler	$10^*$	—	—	ns
42	Tt0P	T0CKI Period	$\frac{T_{CY} + 40}{N}$ §	—	—	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 17-6: TIMER1, TIMER2, AND TIMER3 CLOCK TIMINGS**



**TABLE 17-6: TIMER1, TIMER2, AND TIMER3 CLOCK REQUIREMENTS**

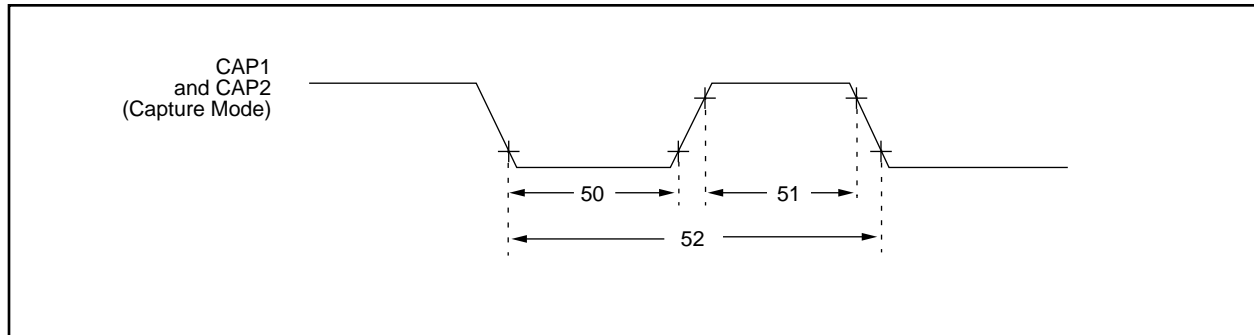
Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
45	Tt123H	TCLK12 and TCLK3 high time	$0.5 T_{CY} + 20$ §	—	—	ns	
46	Tt123L	TCLK12 and TCLK3 low time	$0.5 T_{CY} + 20$ §	—	—	ns	
47	Tt123P	TCLK12 and TCLK3 input period	$\frac{T_{CY} + 40}{N}$ §	—	—	ns	N = prescale value (1, 2, 4, 8)
48	TckE2tmrl	Delay from selected External Clock Edge to Timer increment	$2T_{osc}$ §	—	$6 T_{osc}$ §	—	

\* These parameters are characterized but not tested.

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

§ This specification ensured by design.

**FIGURE 17-7: CAPTURE TIMINGS**



**TABLE 17-7: CAPTURE REQUIREMENTS**

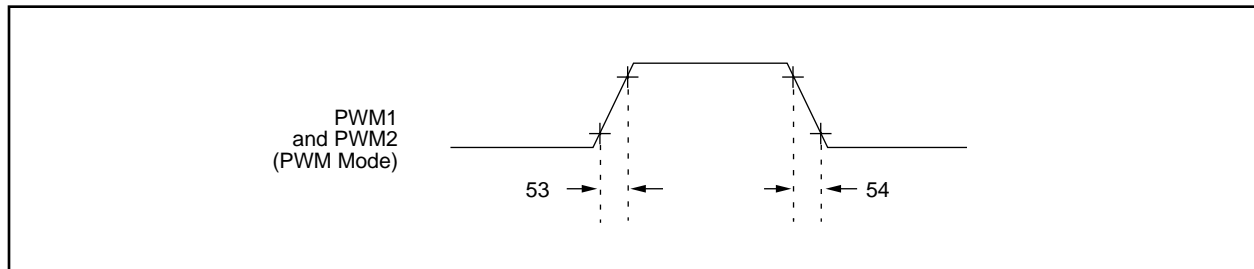
Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
50	TccL	Capture1 and Capture2 input low time	10 *	—	—	ns	
51	TccH	Capture1 and Capture2 input high time	10 *	—	—	ns	
52	TccP	Capture1 and Capture2 input period	$\frac{2 T_{CY}}{N}$ §	—	—	ns	N = prescale value (4 or 16)

\* These parameters are characterized but not tested.

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

§ This specification ensured by design.

**FIGURE 17-8: PWM TIMINGS**



**TABLE 17-8: PWM REQUIREMENTS**

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

\* These parameters are characterized but not tested.

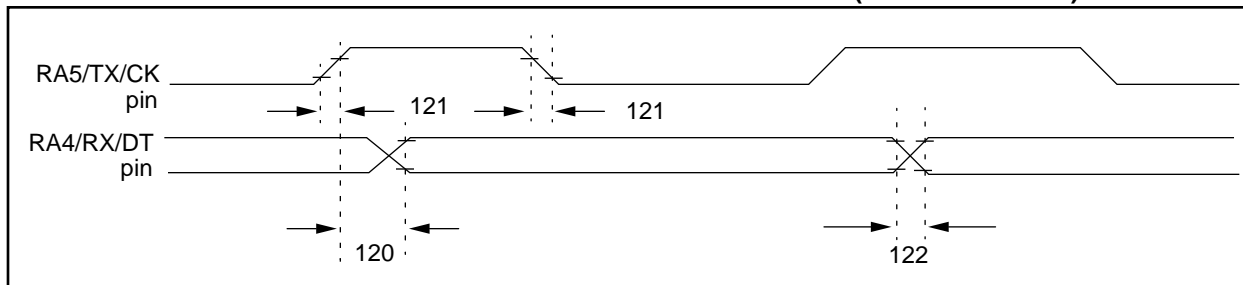
† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

§ This specification ensured by design.

# PIC17C4X

Applicable Devices 42 R42 42A 43 R43 44

**FIGURE 17-9: USART MODULE: SYNCHRONOUS TRANSMISSION (MASTER/SLAVE) TIMING**



**TABLE 17-9: SERIAL PORT SYNCHRONOUS TRANSMISSION REQUIREMENTS**

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
120	TckH2dtV	SYNC XMIT (MASTER & SLAVE) Clock high to data out valid	—	—	65	ns	
121	TckRF	Clock out rise time and fall time (Master Mode)	—	10	35	ns	
122	TdtRF	Data out rise time and fall time	—	10	35	ns	

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**FIGURE 17-10: USART MODULE: SYNCHRONOUS RECEIVE (MASTER/SLAVE) TIMING**



**TABLE 17-10: SERIAL PORT SYNCHRONOUS RECEIVE REQUIREMENTS**

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
125	TdtV2ckL	SYNC RCV (MASTER & SLAVE) Data hold before CK↓ (DT hold time)	15	—	—	ns	
126	TckL2dtl	Data hold after CK↓ (DT hold time)	15	—	—	ns	

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.



**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 WR↓ (data setup time)	0.25Tcy - 40	—	—	ns	
153	TwrH2adI	WR↑ to data out invalid (data hold time)	—	0.25Tcy §	—	ns	
154	TwrL	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.

# PIC17C4X

Applicable Devices 42 R42 42A 43 R43 44

FIGURE 17-12: MEMORY INTERFACE READ TIMING



TABLE 17-12: MEMORY INTERFACE READ REQUIREMENTS

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
150	TadV2aL	AD<15:0> (address) valid to ALE↓ (address setup time)	0.25Tcy - 30	—	—	ns	
151	TalL2adI	ALE↓ to address out invalid (address hold time)	5*	—	—	ns	
160	TadZ2oeL	AD<15:0> high impedance to OE↓	0*	—	—	ns	
161	ToeH2adD	OE↑ to AD<15:0> driven	0.25Tcy - 15	—	—	ns	
162	TadV2oeH	Data in valid before OE↑ (data setup time)	35	—	—	ns	
163	ToeH2adI	OE↑ to data in invalid (data hold time)	0	—	—	ns	
164	TalH	ALE pulse width	—	0.25Tcy §	—	ns	
165	ToeL	OE pulse width	0.5Tcy - 35 §	—	—	ns	
166	TalH2alH	ALE↑ to ALE↑ (cycle time)	—	Tcy §	—	ns	
167	Tacc	Address access time	—	—	0.75 Tcy-40	ns	
168	Toe	Output enable access time (OE low to Data Valid)	—	—	0.5 Tcy - 60	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 guaranteed by design.

## 18.0 PIC17C42 DC AND AC CHARACTERISTICS

The graphs and tables provided in this section are for design guidance and are not tested or guaranteed. In some graphs or tables the data presented are outside specified operating range (e.g. outside specified VDD range). This is for information only and devices are ensured to operate properly only within the specified range.

The data presented in this section is a statistical summary of data collected on units from different lots over a period of time. "Typical" represents the mean of the distribution while "max" or "min" represents (mean + 3σ) and (mean - 3σ) respectively where σ is standard deviation.

**TABLE 18-1: PIN CAPACITANCE PER PACKAGE TYPE**

Pin Name	Typical Capacitance (pF)			
	40-pin DIP	44-pin PLCC	44-pin MQFP	44-pin TQFP
All pins, except $\overline{\text{MCLR}}$ , VDD, and VSS	10	10	10	10
$\overline{\text{MCLR}}$ pin	20	20	20	20

**FIGURE 18-1: TYPICAL RC OSCILLATOR FREQUENCY vs. TEMPERATURE**



# PIC17C4X

Applicable Devices 42 R42 42A 43 R43 44

FIGURE 18-2: TYPICAL RC OSCILLATOR FREQUENCY vs. VDD

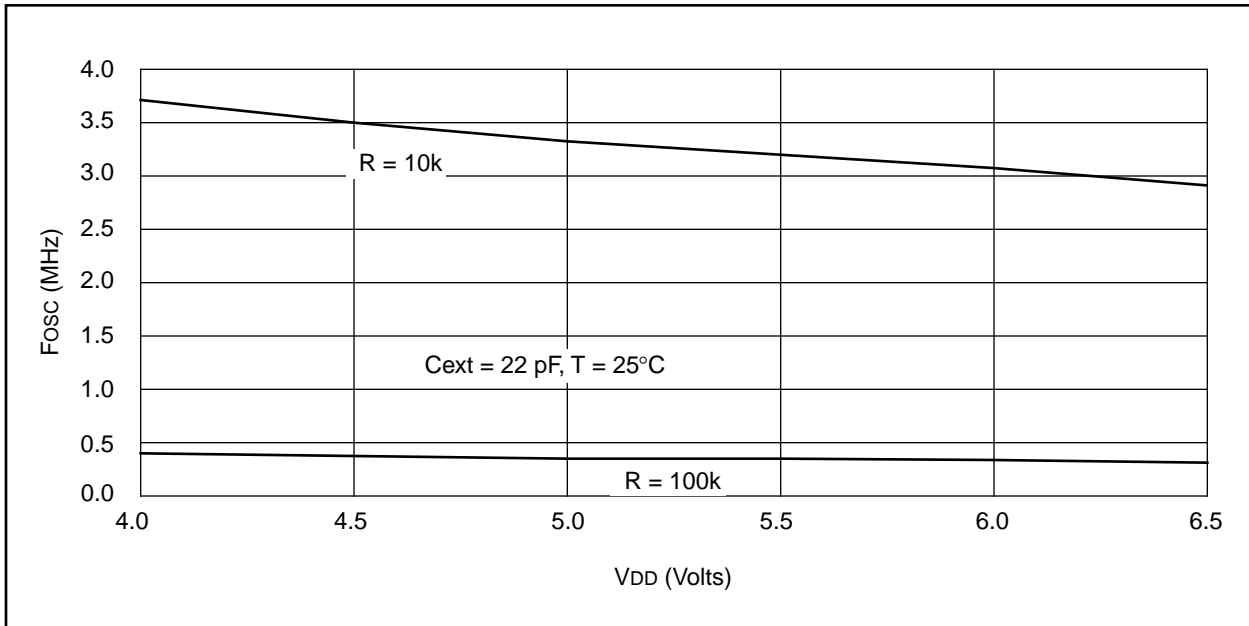
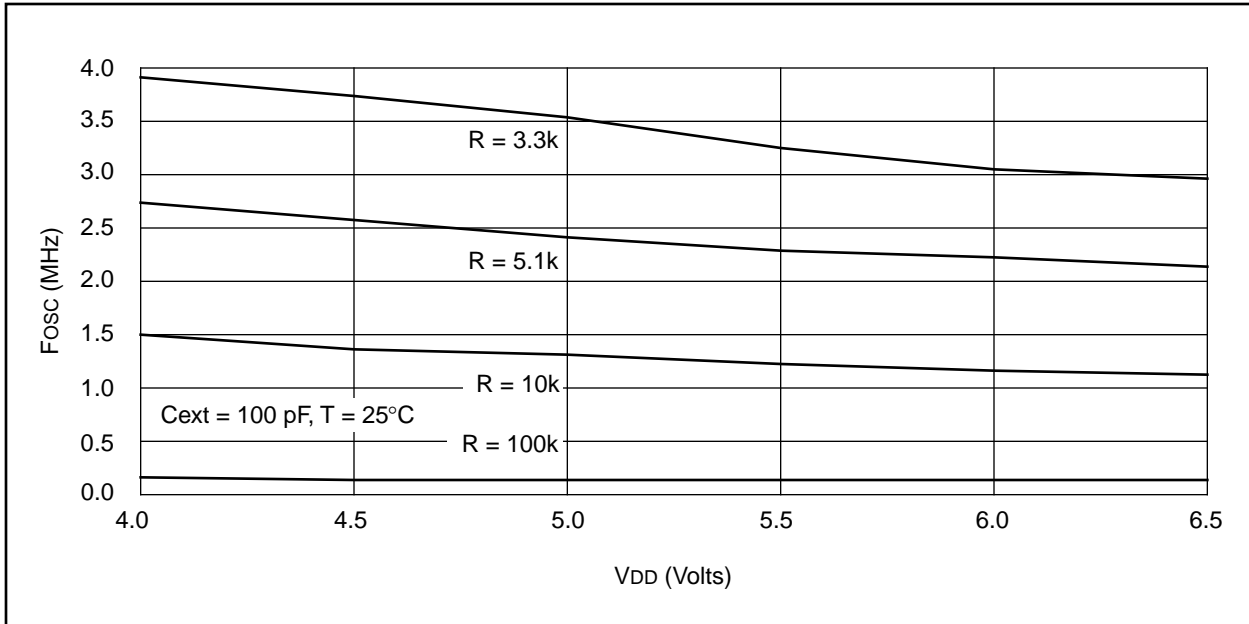


FIGURE 18-3: TYPICAL RC OSCILLATOR FREQUENCY vs. VDD



**FIGURE 18-4: TYPICAL RC OSCILLATOR FREQUENCY vs. VDD**



**TABLE 18-2: RC OSCILLATOR FREQUENCIES**

Cext	Rext	Average Fosc @ 5V, 25°C	
		Fosc (MHz)	Tolerance
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%

# PIC17C4X

Applicable Devices 42 R42 42A 43 R43 44

FIGURE 18-5: TRANSCONDUCTANCE (gm) OF LF OSCILLATOR vs. VDD



FIGURE 18-6: TRANSCONDUCTANCE (gm) OF XT OSCILLATOR vs. VDD



**FIGURE 18-7: TYPICAL  $I_{DD}$  vs. FREQUENCY (EXTERNAL CLOCK 25°C)**



**FIGURE 18-8: MAXIMUM  $I_{DD}$  vs. FREQUENCY (EXTERNAL CLOCK 125°C TO -40°C)**



# PIC17C4X

Applicable Devices 42 R42 42A 43 R43 44

FIGURE 18-9: TYPICAL  $I_{PD}$  vs.  $V_{DD}$  WATCHDOG DISABLED 25°C

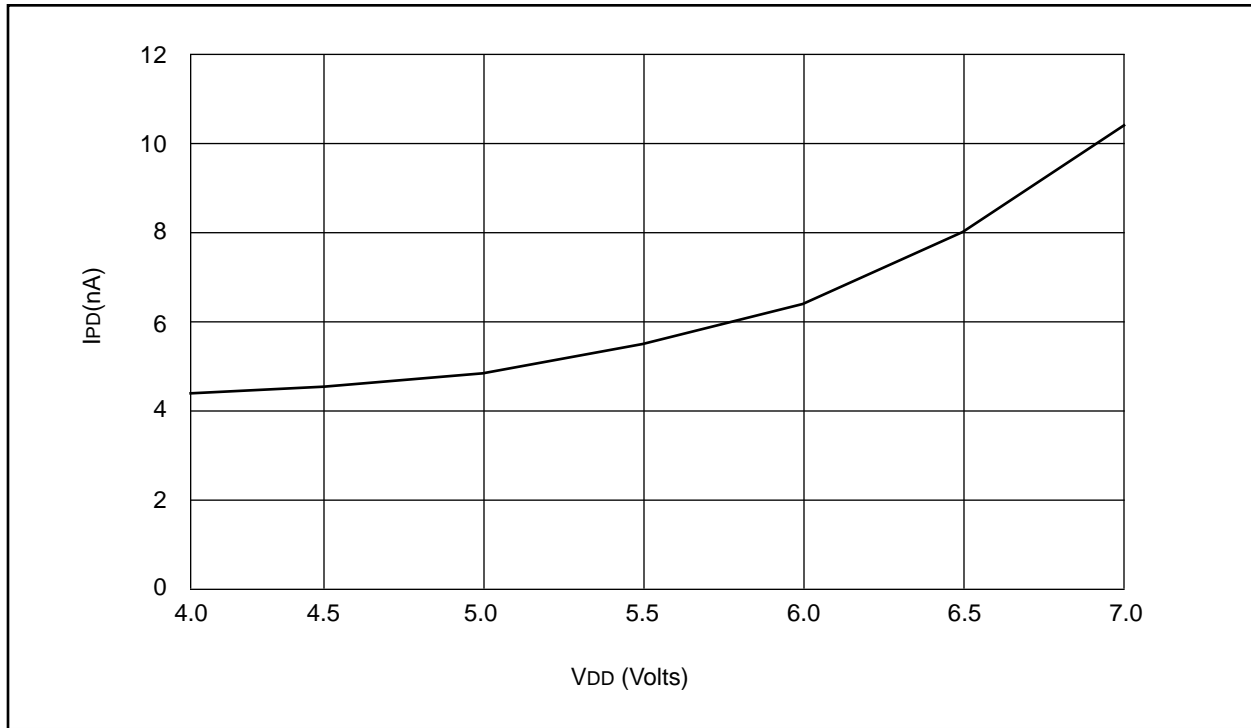
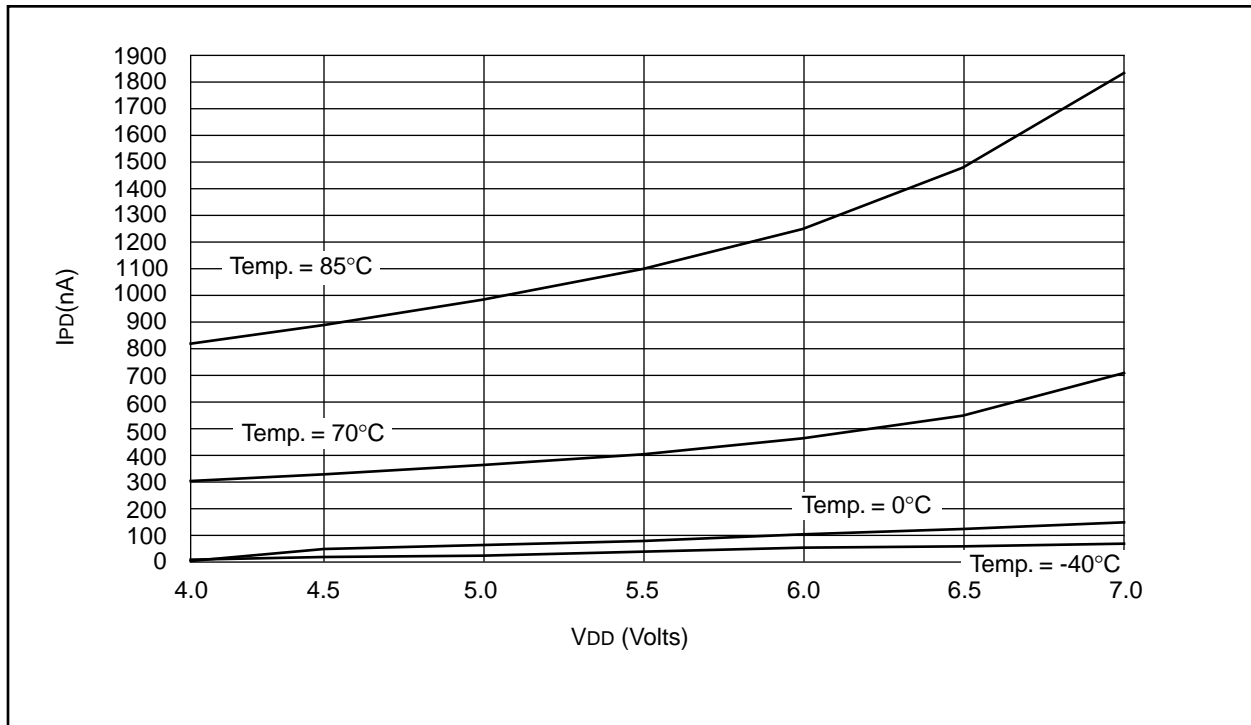
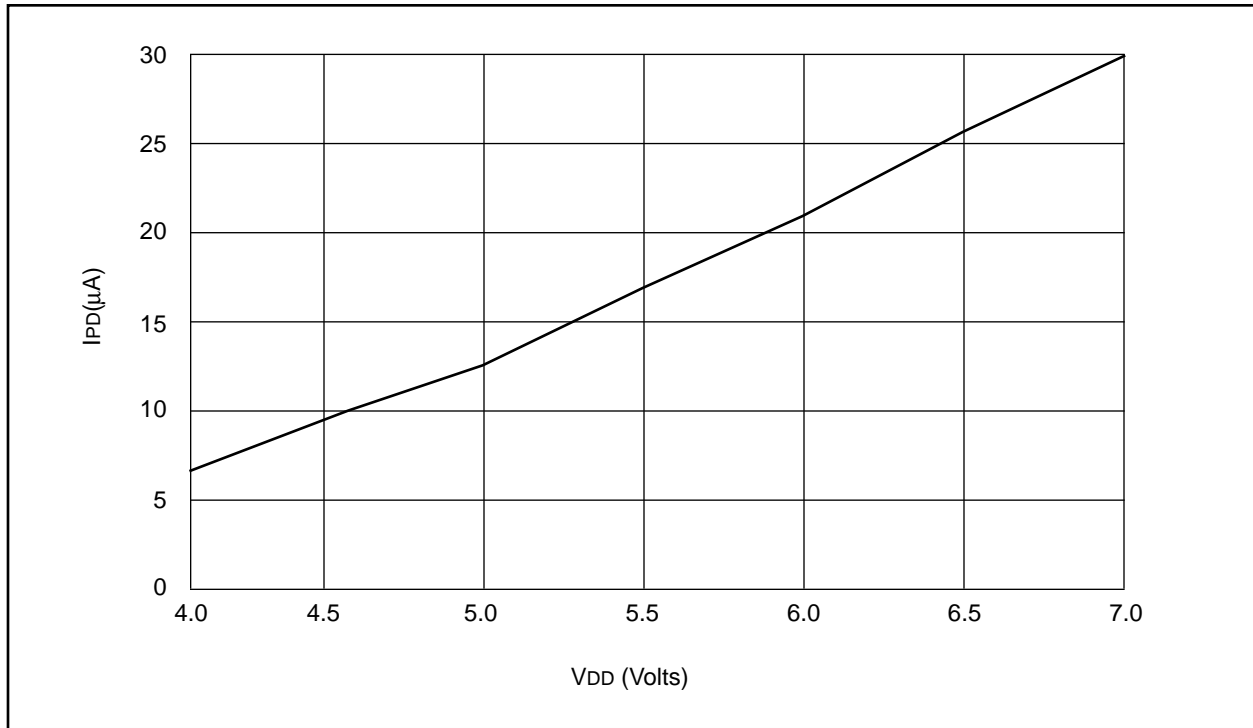


FIGURE 18-10: MAXIMUM  $I_{PD}$  vs.  $V_{DD}$  WATCHDOG DISABLED

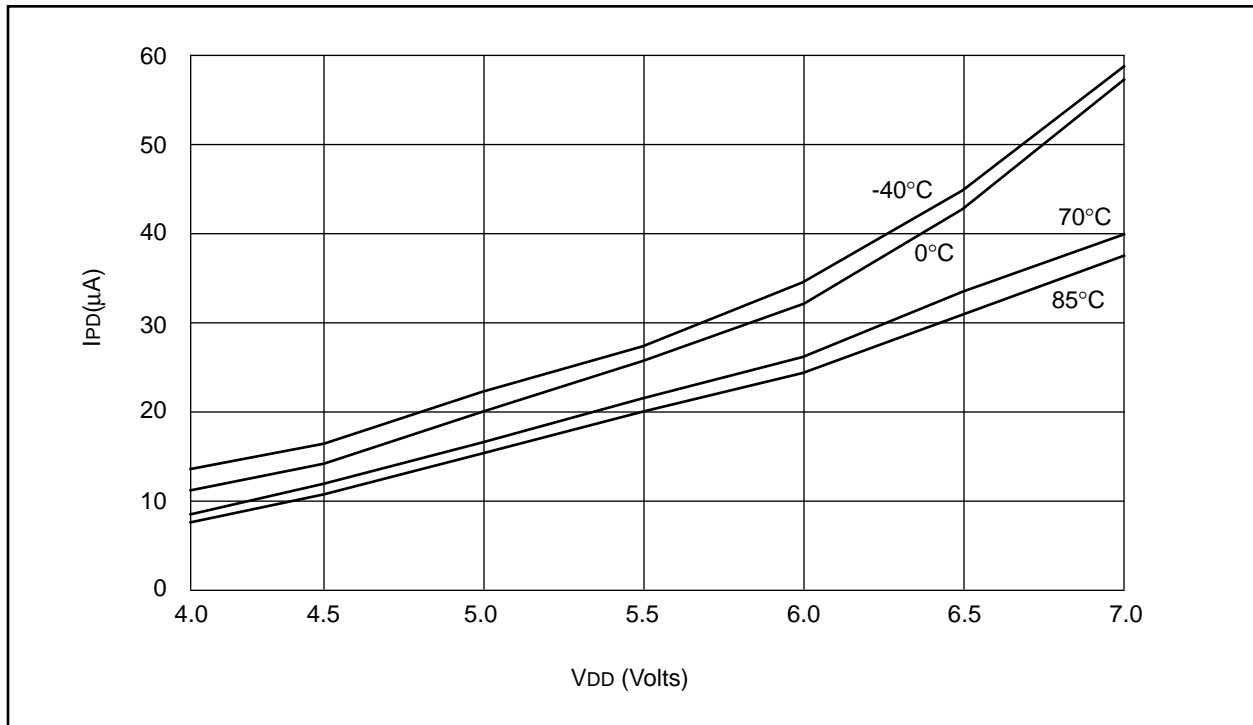




**FIGURE 18-11: TYPICAL  $I_{PD}$  vs.  $V_{DD}$  WATCHDOG ENABLED 25°C**



**FIGURE 18-12: MAXIMUM  $I_{PD}$  vs.  $V_{DD}$  WATCHDOG ENABLED**



# PIC17C4X

Applicable Devices 42 R42 42A 43 R43 44

FIGURE 18-13: WDT TIMER TIME-OUT PERIOD vs. VDD



FIGURE 18-14: I<sub>OH</sub> vs. V<sub>OH</sub>, VDD = 3V



**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 18-19:  $V_{IH}$ ,  $V_{IL}$  of I/O PINS (SCHMITT TRIGGER) vs.  $V_{DD}$**



**FIGURE 18-20:  $V_{TH}$  (INPUT THRESHOLD VOLTAGE) OF OSC1 INPUT (IN XT AND LF MODES) vs.  $V_{DD}$**



# PIC17C4X

---

NOTES:

## 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 VDD with respect to VSS .....	0 to +7.5V
Voltage on $\overline{\text{MCLR}}$ with respect to VSS (Note 2) .....	-0.6V to +14V
Voltage on RA2 and RA3 with respect to VSS.....	-0.6V to +14V
Voltage on all other pins with respect to VSS .....	-0.6V to VDD + 0.6V
Total power dissipation (Note 1).....	1.0W
Maximum current out of VSS pin(s) - total .....	250 mA
Maximum current into VDD pin(s) - total .....	200 mA
Input clamp current, I <sub>IK</sub> (V <sub>I</sub> < 0 or V <sub>I</sub> > VDD) .....	±20 mA
Output clamp current, I <sub>OK</sub> (V <sub>O</sub> < 0 or V <sub>O</sub> > VDD).....	±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:  $P_{dis} = V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD} - V_{OH}) \times I_{OH}\} + \sum (V_{OL} \times I_{OL})$

**Note 2:** Voltage spikes below VSS at the  $\overline{\text{MCLR}}$  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  $\overline{\text{MCLR}}$  pin rather than pulling this pin directly to VSS.

† 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

Applicable Devices 42 R42 42A 43 R43 44

**TABLE 19-1: CROSS REFERENCE OF DEVICE SPECS FOR OSCILLATOR CONFIGURATIONS AND FREQUENCIES OF OPERATION (COMMERCIAL DEVICES)**

	PIC17LCR42-08 PIC17LC42A-08 PIC17LC43-08 PIC17LCR43-08 PIC17LC44-08	PIC17CR42-16 PIC17C42A-16 PIC17C43-16 PIC17CR43-16 PIC17C44-16	PIC17CR42-25 PIC17C42A-25 PIC17C43-25 PIC17CR43-25 PIC17C44-25	PIC17CR42-33 PIC17C42A-33 PIC17C43-33 PIC17CR43-33 PIC17C44-33	JW Devices (Ceramic Windowed Devices)
OSC	VDD: 2.5V to 6.0V IDD: 6 mA max. IPD: 5 µA max. at 5.5V WDT disabled Freq: 4 MHz max.	VDD: 4.5V to 6.0V IDD: 6 mA max. IPD: 5 µA max. at 5.5V WDT disabled Freq: 4 MHz max.	VDD: 4.5V to 6.0V IDD: 6 mA max. IPD: 5 µA max. at 5.5V WDT disabled Freq: 4 MHz max.	VDD: 4.5V to 6.0V IDD: 6 mA max. IPD: 5 µA max. at 5.5V WDT disabled Freq: 4 MHz max.	VDD: 4.5V to 6.0V IDD: 6 mA max. IPD: 5 µA max. at 5.5V WDT disabled Freq: 4 MHz max.
RC	VDD: 2.5V to 6.0V IDD: 12 mA max. IPD: 5 µA max. at 5.5V WDT disabled Freq: 8 MHz max.	VDD: 4.5V to 6.0V IDD: 24 mA max. IPD: 5 µA max. at 5.5V WDT disabled Freq: 16 MHz max.	VDD: 4.5V to 6.0V IDD: 38 mA max. IPD: 5 µA max. at 5.5V WDT disabled Freq: 25 MHz max.	VDD: 4.5V to 6.0V IDD: 38 mA max. IPD: 5 µA max. at 5.5V WDT disabled Freq: 33 MHz max.	VDD: 4.5V to 6.0V IDD: 38 mA max. IPD: 5 µA max. at 5.5V WDT disabled Freq: 33 MHz max.
XT	VDD: 2.5V to 6.0V IDD: 12 mA max. IPD: 5 µA max. at 5.5V WDT disabled Freq: 8 MHz max.	VDD: 4.5V to 6.0V IDD: 24 mA max. IPD: 5 µA max. at 5.5V WDT disabled Freq: 16 MHz max.	VDD: 4.5V to 6.0V IDD: 38 mA max. IPD: 5 µA max. at 5.5V WDT disabled Freq: 25 MHz max.	VDD: 4.5V to 6.0V IDD: 38 mA max. IPD: 5 µA max. at 5.5V WDT disabled Freq: 33 MHz max.	VDD: 4.5V to 6.0V IDD: 38 mA max. IPD: 5 µA max. at 5.5V WDT disabled Freq: 33 MHz max.
EC	VDD: 2.5V to 6.0V IDD: 12 mA max. IPD: 5 µA max. at 5.5V WDT disabled Freq: 8 MHz max.	VDD: 4.5V to 6.0V IDD: 24 mA max. IPD: 5 µA max. at 5.5V WDT disabled Freq: 16 MHz Max	VDD: 4.5V to 6.0V IDD: 38 mA max. IPD: 5 µA max. at 5.5V WDT disabled Freq: 25 MHz max.	VDD: 4.5V to 6.0V IDD: 38 mA max. IPD: 5 µA max. at 5.5V WDT disabled Freq: 33 MHz max.	VDD: 4.5V to 6.0V IDD: 38 mA max. IPD: 5 µA max. at 5.5V WDT disabled Freq: 33 MHz max.
LF	VDD: 2.5V to 6.0V IDD: 150 µA max. at 32 kHz IPD: 5 µA max. at 5.5V WDT disabled Freq: 2 MHz max.	VDD: 4.5V to 6.0V IDD: 95 µA typ. at 32 kHz IPD: < 1 µA typ. at 5.5V WDT disabled Freq: 2 MHz max.	VDD: 4.5V to 6.0V IDD: 95 µA typ. at 32 kHz IPD: < 1 µA typ. at 5.5V WDT disabled Freq: 2 MHz max.	VDD: 4.5V to 6.0V IDD: 95 µA typ. at 32 kHz IPD: < 1 µA typ. at 5.5V WDT disabled Freq: 2 MHz max.	VDD: 2.5V to 6.0V IDD: 150 µA max. at 32 kHz IPD: 5 µA max. at 5.5V WDT disabled Freq: 2 MHz max.

The shaded sections indicate oscillator selections which are tested for functionality, but not for MIN/MAX specifications. It is recommended that the user select the device type that ensures the specifications required.



**19.1 DC CHARACTERISTICS:**      **PIC17CR42/42A/43/R43/44-16 (Commercial, Industrial)**  
**PIC17CR42/42A/43/R43/44-25 (Commercial, Industrial)**  
**PIC17CR42/42A/43/R43/44-33 (Commercial, Industrial)**

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					
Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
D001	VDD	Supply Voltage	4.5	–	6.0	V	
D002	VDR	RAM Data Retention Voltage (Note 1)	1.5 *	–	–	V	Device in SLEEP mode
D003	VPOR	VDD start voltage to ensure internal Power-on Reset signal	–	VSS	–	V	See section on Power-on Reset for details
D004	SvDD	VDD rise rate to ensure internal Power-on Reset signal	0.060 *	–	–	mV/ms	See section on Power-on Reset for details
D010	IDD	Supply Current (Note 2)	–	3	6	mA	FOSC = 4 MHz (Note 4)
D011			–	6	12 *	mA	FOSC = 8 MHz
D012			–	11	24 *	mA	FOSC = 16 MHz
D013			–	19	38	mA	FOSC = 25 MHz
D015			–	25	50	mA	FOSC = 33 MHz
D014			–	95	150	µA	FOSC = 32 kHz, WDT enabled (EC osc configuration)
D020	IPD	Power-down Current (Note 3)	–	10	40	µA	VDD = 5.5V, WDT enabled
D021			–	< 1	5	µA	VDD = 5.5V, WDT disabled

\* 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.

Note 1: This is the limit to which VDD can be lowered in SLEEP mode without losing RAM data.

2: The supply current is mainly a function of the operating voltage and frequency. Other factors such as I/O pin loading and switching rate, oscillator type, internal code execution pattern, and temperature also have an impact on the current consumption.

The test conditions for all IDD measurements in active operation mode are:

OSC1 = external square wave, from rail to rail; all I/O pins tristated, pulled to VDD or VSS, T0CKI = VDD, MCLR = VDD; WDT enabled/disabled as specified.

Current consumed from the oscillator and I/O's driving external capacitive or resistive loads needs to be considered.

For the RC oscillator, the current through the external pull-up resistor (R) can be estimated as:  $V_{DD} / (2 \cdot R)$ .

For capacitive loads, the current can be estimated (for an individual I/O pin) as  $(CL \cdot V_{DD}) \cdot f$

CL = Total capacitive load on the I/O pin; f = average frequency the I/O pin switches.

The capacitive currents are most significant when the device is configured for external execution (includes extended microcontroller mode).

3: The power down current in SLEEP mode does not depend on the oscillator type. Power-down current is measured with the part in SLEEP mode, with all I/O pins in hi-impedance state and tied to VDD and VSS.

4: For RC osc configuration, current through Rext is not included. The current through the resistor can be estimated by the formula  $I_R = V_{DD}/2R_{ext}$  (mA) with Rext in kOhm.

# PIC17C4X

Applicable Devices 42 R42 42A 43 R43 44

## 19.2 DC CHARACTERISTICS: PIC17LC42A/43/LC44 (Commercial, Industrial) PIC17LCR42/43 (Commercial, Industrial)

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					
Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
D001	VDD	Supply Voltage	2.5	–	6.0	V	
D002	VDR	RAM Data Retention Voltage (Note 1)	1.5 *	–	–	V	Device in SLEEP mode
D003	VPOR	VDD start voltage to ensure internal Power-on Reset signal	–	VSS	–	V	See section on Power-on Reset for details
D004	SVDD	VDD rise rate to ensure internal Power-on Reset signal	0.060 *	–	–	mV/ms	See section on Power-on Reset for details
D010 D011 D014	IDD	Supply Current (Note 2)	–	3 6 95	6 12 * 150	mA mA μA	FOSC = 4 MHz (Note 4) FOSC = 8 MHz FOSC = 32 kHz, WDT disabled (EC osc configuration)
D020 D021	IPD	Power-down Current (Note 3)	–	10 < 1	40 5	μA μA	VDD = 5.5V, WDT enabled VDD = 5.5V, WDT disabled

\* 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.

Note 1: This is the limit to which VDD can be lowered in SLEEP mode without losing RAM data.

2: The supply current is mainly a function of the operating voltage and frequency. Other factors such as I/O pin loading and switching rate, oscillator type, internal code execution pattern, and temperature also have an impact on the current consumption.

The test conditions for all IDD measurements in active operation mode are:

OSC1=external square wave, from rail to rail; all I/O pins tristated, pulled to VDD or VSS, T0CKI = VDD, MCLR = VDD; WDT enabled/disabled as specified.

Current consumed from the oscillator and I/O's driving external capacitive or resistive loads needs to be considered.

For the RC oscillator, the current through the external pull-up resistor (R) can be estimated as:  $V_{DD} / (2 \cdot R)$ .

For capacitive loads, the current can be estimated (for an individual I/O pin) as  $(C_L \cdot V_{DD}) \cdot f$

CL = Total capacitive load on the I/O pin; f = average frequency the I/O pin switches.

The capacitive currents are most significant when the device is configured for external execution (includes extended microcontroller mode).

3: The power down current in SLEEP mode does not depend on the oscillator type. Power-down current is measured with the part in SLEEP mode, with all I/O pins in hi-impedance state and tied to VDD or VSS.

4: For RC osc configuration, current through Rext is not included. The current through the resistor can be estimated by the formula  $I_R = V_{DD}/2R_{ext}$  (mA) with Rext in kOhm.

# PIC17C4X

Applicable Devices 42 R42 42A 43 R43 44

**19.3 DC CHARACTERISTICS:** PIC17CR42/42A/43/R43/44-16 (Commercial, Industrial)  
 PIC17CR42/42A/43/R43/44-25 (Commercial, Industrial)  
 PIC17CR42/42A/43/R43/44-33 (Commercial, Industrial)  
 PIC17LCR42/42A/43/R43/44-08 (Commercial, Industrial)

Standard Operating Conditions (unless otherwise stated)							
DC CHARACTERISTICS							
Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial and $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$ for commercial							
Operating voltage VDD range as described in Section 19.1							
Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
<b>Input Low Voltage</b>							
D030	VIL	I/O ports	VSS	–	0.8	V	$4.5\text{V} \leq V_{DD} \leq 5.5\text{V}$ $2.5\text{V} \leq V_{DD} \leq 4.5\text{V}$
		with TTL buffer	VSS	–	0.2VDD	V	
D031		with Schmitt Trigger buffer	VSS	–	0.2VDD	V	
D032		MCLR, OSC1 (in EC and RC mode)	VSS	–	0.2VDD	V	Note1
D033		OSC1 (in XT, and LF mode)	–	0.5VDD	–	V	
<b>Input High Voltage</b>							
D040	VIH	I/O ports	2.0	–	VDD	V	$4.5\text{V} \leq V_{DD} \leq 5.5\text{V}$ $2.5\text{V} \leq V_{DD} \leq 4.5\text{V}$
		with TTL buffer	$1 + 0.2V_{DD}$	–	VDD	V	
D041		with Schmitt Trigger buffer	0.8VDD	–	VDD	V	
D042		MCLR	0.8VDD	–	VDD	V	Note1
D043		OSC1 (XT, and LF mode)	–	0.5VDD	–	V	
D050	VHYS	Hysteresis of Schmitt Trigger inputs	$0.15V_{DD}^*$	–	–	V	
<b>Input Leakage Current</b> (Notes 2, 3)							
D060	IIL	I/O ports (except RA2, RA3)	–	–	$\pm 1$	$\mu\text{A}$	$V_{SS} \leq V_{PIN} \leq V_{DD}$ , I/O Pin at hi-impedance PORTB weak pull-ups disabled
D061		MCLR	–	–	$\pm 2$	$\mu\text{A}$	$V_{PIN} = V_{SS}$ or $V_{PIN} = V_{DD}$
D062		RA2, RA3	–	–	$\pm 2$	$\mu\text{A}$	$V_{SS} \leq V_{RA2}, V_{RA3} \leq 12\text{V}$
D063		OSC1, TEST (EC, RC modes)	–	–	$\pm 1$	$\mu\text{A}$	$V_{SS} \leq V_{PIN} \leq V_{DD}$
D063B		OSC1, TEST (XT, LF modes)	–	–	V <sub>PIN</sub>	$\mu\text{A}$	$R_F \geq 1\text{M}\Omega$ , see Figure 14.2
D064		MCLR	–	–	10	$\mu\text{A}$	$V_{MCLR} = V_{PP} = 12\text{V}$ (when not programming)
D070	IPURB	PORTB weak pull-up current	60	200	400	$\mu\text{A}$	$V_{PIN} = V_{SS}$ , $\overline{RBPU} = 0$ $4.5\text{V} \leq V_{DD} \leq 6.0\text{V}$

\* 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 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 MCLR/VPP 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.

# PIC17C4X

Applicable Devices 42 R42 42A 43 R43 44

Standard Operating Conditions (unless otherwise stated)							
DC CHARACTERISTICS							
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
D080	VOL	<b>Output Low Voltage</b> I/O ports (except RA2 and RA3)	–	–	0.1VDD	V	IO <sub>L</sub> = VDD/1.250 mA 4.5V ≤ VDD ≤ 6.0V
D081		with TTL buffer	–	–	0.1VDD * 0.4	V	VDD = 2.5V IO <sub>L</sub> = 6 mA, VDD = 4.5V Note 6
D082		RA2 and RA3	–	–	3.0	V	IO <sub>L</sub> = 60.0 mA, VDD = 6.0V
D083		OSC2/CLKOUT	–	–	0.4	V	IO <sub>L</sub> = 1 mA, VDD = 4.5V
D084		(RC and EC osc modes)	–	–	0.1VDD *	V	IO <sub>L</sub> = VDD/5 mA (PIC17LC43/LC44 only)
D090	VOH	<b>Output High Voltage (Note 3)</b> I/O ports (except RA2 and RA3)	0.9VDD 0.9VDD *	–	–	V	IO <sub>H</sub> = -VDD/2.500 mA 4.5V ≤ VDD ≤ 6.0V
D091		with TTL buffer	2.4	–	–	V	VDD = 2.5V IO <sub>H</sub> = -6.0 mA, VDD=4.5V Note 6
D092		RA2 and RA3	–	–	12	V	Pulled-up to externally applied voltage
D093		OSC2/CLKOUT	2.4	–	–	V	IO <sub>H</sub> = -5 mA, VDD = 4.5V
D094		(RC and EC osc modes)	0.9VDD *	–	–	V	IO <sub>H</sub> = -VDD/5 mA (PIC17LC43/LC44 only)
D100	Cosc2	<b>Capacitive Loading Specs on Output Pins</b> 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	CIO	All I/O pins and OSC2 (in RC mode)	–	–	50	pF	
D102	CAD	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 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 MCLR/VPP 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.

# PIC17C4X

Applicable Devices	42	R42	42A	43	R43	44
--------------------	----	-----	-----	----	-----	----

DC CHARACTERISTICS		Standard Operating Conditions (unless otherwise stated)					
		Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +40^{\circ}\text{C}$					
		Operating voltage VDD range as described in Section 19.1					
Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
		<b>Internal Program Memory Programming Specs (Note 4)</b>					
D110	VPP	Voltage on $\overline{\text{MCLR}}/\text{VPP}$ pin	12.75	–	13.25	V	Note 5
D111	VDDP	Supply voltage during programming	4.75	5.0	5.25	V	
D112	I <sub>PP</sub>	Current into $\overline{\text{MCLR}}/\text{VPP}$ pin	–	25 ‡	50 ‡	mA	
D113	I <sub>DDP</sub>	Supply current during programming	–	–	30 ‡	mA	
D114	T <sub>PROG</sub>	Programming pulse width	10	100	1000	μs	Terminated via internal/external interrupt or a reset

\* 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  $\overline{\text{MCLR}}$  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  $\overline{\text{MCLR}}/\text{VPP}$  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.

<b>Note:</b> When using the Table Write for internal programming, the device temperature must be less than 40°C.
--

# PIC17C4X

Applicable Devices 42 R42 42A 43 R43 44

## 19.4 Timing Parameter Symbology

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

1. TppS2ppS
2. TppS
3. TCC:ST (I<sup>2</sup>C specifications only)
4. Ts (I<sup>2</sup>C specifications only)

<b>T</b>	
F	Frequency
T	Time

Lowercase symbols (pp) and their meanings:

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

Uppercase symbols and their meanings:

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

**FIGURE 19-1: PARAMETER MEASUREMENT INFORMATION**

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



# PIC17C4X

Applicable Devices 42 R42 42A 43 R43 44

## 19.5 Timing Diagrams and Specifications

FIGURE 19-2: EXTERNAL CLOCK TIMING

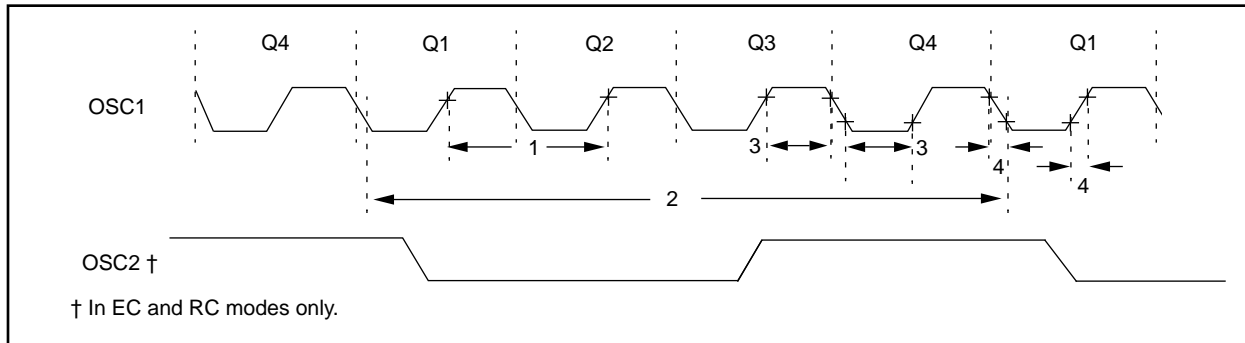


TABLE 19-2: EXTERNAL CLOCK TIMING REQUIREMENTS

Param No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
	Fosc	External CLKIN Frequency (Note 1)	DC	—	8	MHz	EC osc mode - 08 devices (8 MHz devices) - 16 devices (16 MHz devices) - 25 devices (25 MHz devices) - 33 devices (33 MHz devices)
			DC	—	16	MHz	
			DC	—	25	MHz	
			DC	—	33	MHz	
		Oscillator Frequency (Note 1)	DC	—	4	MHz	RC osc mode XT osc mode - 08 devices (8 MHz devices) - 16 devices (16 MHz devices) - 25 devices (25 MHz devices) - 33 devices (33 MHz devices) LF osc mode
1	—	8	MHz				
1	—	16	MHz				
1	—	25	MHz				
1	Tosc	External CLKIN Period (Note 1)	125	—	—	ns	EC osc mode - 08 devices (8 MHz devices) - 16 devices (16 MHz devices) - 25 devices (25 MHz devices) - 33 devices (33 MHz devices)
			62.5	—	—	ns	
			40	—	—	ns	
			30.3	—	—	ns	
		Oscillator Period (Note 1)	250	—	—	ns	RC osc mode XT osc mode - 08 devices (8 MHz devices) - 16 devices (16 MHz devices) - 25 devices (25 MHz devices) - 33 devices (33 MHz devices) LF osc mode
125	—	1,000	ns				
62.5	—	1,000	ns				
40	—	1,000	ns				
30.3	—	1,000	ns				
500	—	—	ns				
2	Tcy	Instruction Cycle Time (Note 1)	121.2	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 an 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/CLKIN pin. When an external clock input is used, the "max." cycle time limit is "DC" (no clock) for all devices.



**FIGURE 19-3: CLKOUT AND I/O TIMING**



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

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions	
10	TosH2ckL	OSC1↓ to CLKOUT↓	—	15 ‡	30 ‡	ns	Note 1	
11	TosH2ckH	OSC1↓ to CLKOUT↑	—	15 ‡	30 ‡	ns	Note 1	
12	TckR	CLKOUT rise time	—	5 ‡	15 ‡	ns	Note 1	
13	TckF	CLKOUT fall time	—	5 ‡	15 ‡	ns	Note 1	
14	TckH2ioV	CLKOUT ↑ to Port out valid	PIC17CR42/42A/43/R43/44	—	—	0.5Tcy + 20 ‡	ns	Note 1
			PIC17LCR42/42A/43/R43/44	—	—	0.5Tcy + 50 ‡	ns	Note 1
15	TioV2ckH	Port in valid before CLKOUT↑	PIC17CR42/42A/43/R43/44	0.25Tcy + 25 ‡	—	—	ns	Note 1
			PIC17LCR42/42A/43/R43/44	0.25Tcy + 50 ‡	—	—	ns	Note 1
16	TckH2ioL	Port in hold after CLKOUT↑	0 ‡	—	—	ns	Note 1	
17	TosH2ioV	OSC1↓ (Q1 cycle) to Port out valid	—	—	100 ‡	ns		
18	TosH2ioL	OSC1↓ (Q2 cycle) to Port input invalid (I/O in hold time)	0 ‡	—	—	ns		
19	TioV2osH	Port input valid to OSC1↓ (I/O in setup time)	30 ‡	—	—	ns		
20	TioR	Port output rise time	—	10 ‡	35 ‡	ns		
21	TioF	Port output fall time	—	10 ‡	35 ‡	ns		
22	TinHL	INT pin high or low time	25 *	—	—	ns		
23	TrbHL	RB7:RB0 change INT high or low time	25 *	—	—	ns		

\* These parameters are characterized but not tested.

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

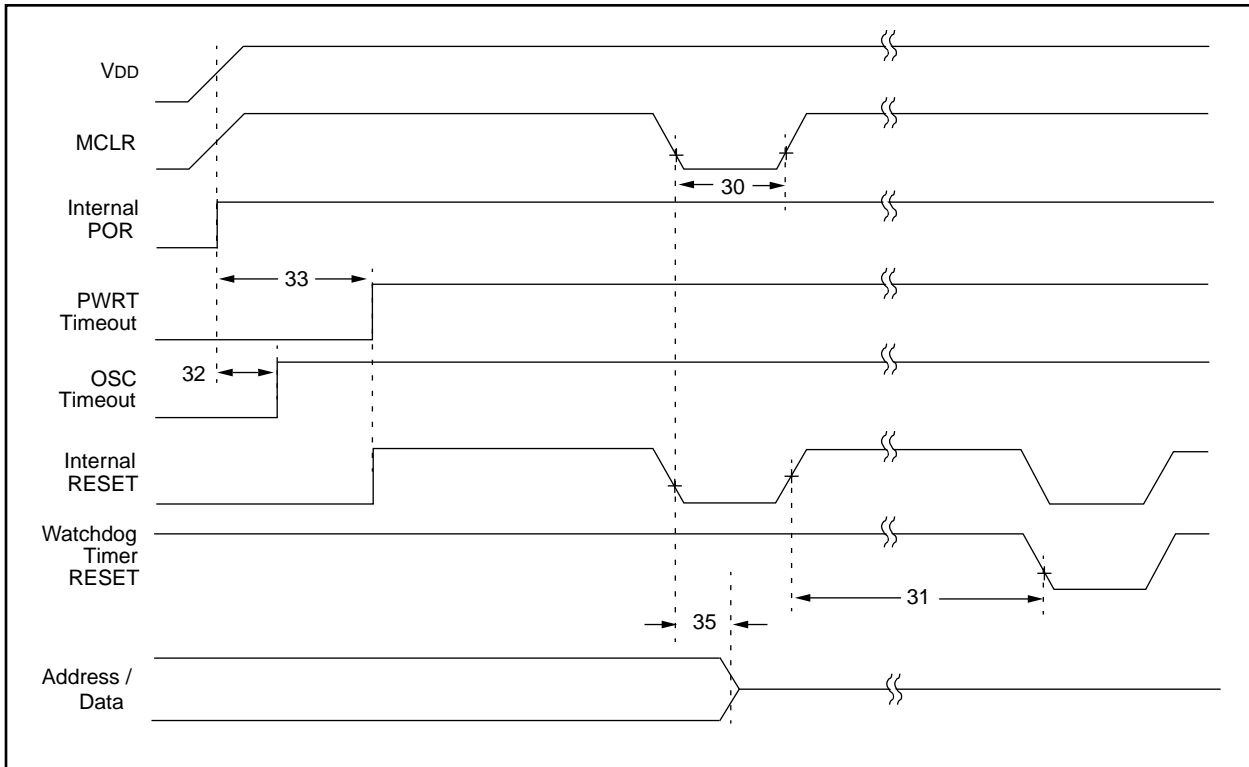
‡ These parameters are for design guidance only and are not tested, nor characterized.

Note 1: Measurements are taken in EC Mode where CLKOUT output is 4 x TOSC.

# PIC17C4X

Applicable Devices 42 R42 42A 43 R43 44

**FIGURE 19-4: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER, AND POWER-UP TIMER TIMING**



**TABLE 19-4: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER REQUIREMENTS**

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions	
30	Tmcl	MCLR Pulse Width (low)	100 *	—	—	ns	VDD = 5V	
31	Twdt	Watchdog Timer Time-out Period (Prescale = 1)	5 *	12	25 *	ms	VDD = 5V	
32	Tost	Oscillation Start-up Timer Period	—	1024Tosc‡	—	ms	Tosc = OSC1 period	
33	Tpwrt	Power-up Timer Period	40 *	96	200 *	ms	VDD = 5V	
35	Tmcl2ad‡	MCLR to System Interface bus (AD15:AD0>) invalid	PIC17CR42/42A/43/R43/44	—	—	100 *	ns	
			PIC17LCR42/42A/43/R43/44	—	—	120 *	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.

§ This specification ensured by design.

**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	$0.5T_{CY} + 20 \text{ §}$	—	—	ns
		With Prescaler	10*	—	—	ns	
41	Tt0L	T0CKI Low Pulse Width	No Prescaler	$0.5T_{CY} + 20 \text{ §}$	—	—	ns
		With Prescaler	10*	—	—	ns	
42	Tt0P	T0CKI Period	Greater of: $20 \text{ ns}$ or $\frac{T_{CY} + 40 \text{ §}}{N}$	—	—	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	$0.5T_{CY} + 20 \text{ §}$	—	—	ns	
46	Tt123L	TCLK12 and TCLK3 low time	$0.5T_{CY} + 20 \text{ §}$	—	—	ns	
47	Tt123P	TCLK12 and TCLK3 input period	$\frac{T_{CY} + 40 \text{ §}}{N}$	—	—	ns	N = prescale value (1, 2, 4, 8)
48	TckE2tmr1	Delay from selected External Clock Edge to Timer increment	$2T_{OSC} \text{ §}$		$6T_{OSC} \text{ §}$		

\* 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 19-7: CAPTURE TIMINGS**



**TABLE 19-7: CAPTURE REQUIREMENTS**

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

\* These parameters are characterized but not tested.

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

§ This specification ensured by design.

**FIGURE 19-8: PWM TIMINGS**



**TABLE 19-8: PWM REQUIREMENTS**

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

\* These parameters are characterized but not tested.

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

§ This specification ensured by design.

**FIGURE 19-9: USART MODULE: SYNCHRONOUS TRANSMISSION (MASTER/SLAVE) TIMING**



**TABLE 19-9: SYNCHRONOUS TRANSMISSION REQUIREMENTS**

Param No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions	
120	TckH2dtV	SYNC XMIT (MASTER & SLAVE)	—	—	50	ns		
		Clock high to data out valid						
121	TckRF	Clock out rise time and fall time (Master Mode)	PIC17CR42/42A/43/R43/44	—	—	25	ns	
			PIC17LCR42/42A/43/R43/44	—	—	40	ns	
122	TdtRF	Data out rise time and fall time	PIC17CR42/42A/43/R43/44	—	—	25	ns	
			PIC17LCR42/42A/43/R43/44	—	—	40	ns	

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**FIGURE 19-10: USART MODULE: SYNCHRONOUS RECEIVE (MASTER/SLAVE) TIMING**



**TABLE 19-10: SYNCHRONOUS RECEIVE REQUIREMENTS**

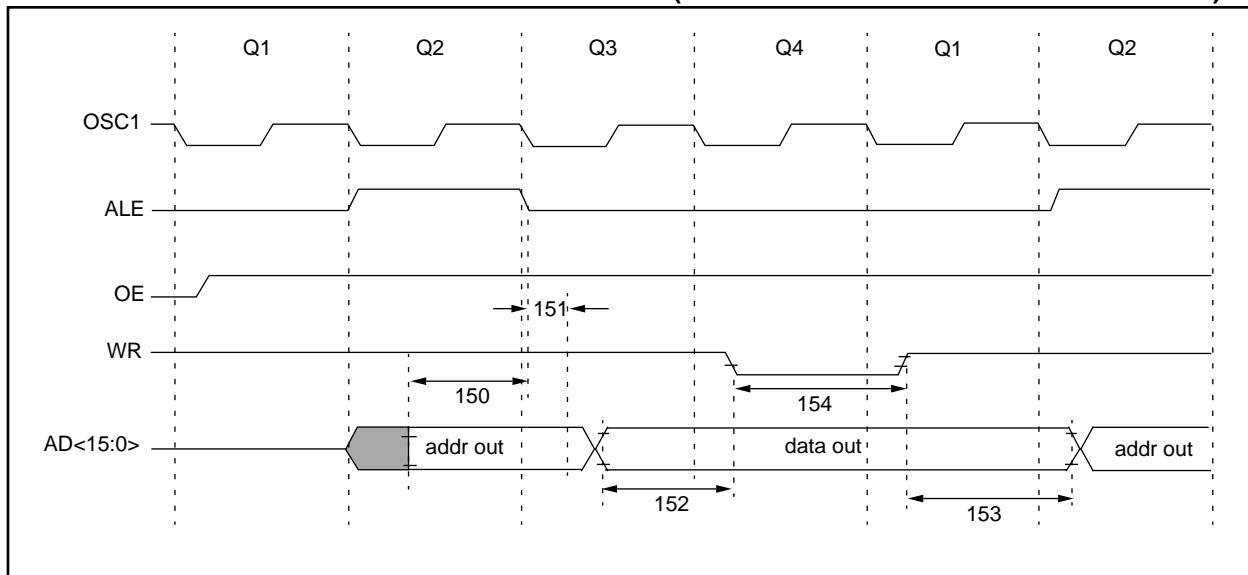
Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
125	TdtV2ckL	SYNC RCV (MASTER & SLAVE) Data hold before CK↓ (DT hold time)	15	—	—	ns	
126	TckL2dtl	Data hold after CK↓ (DT hold time)	15	—	—	ns	

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

# PIC17C4X

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	TadV2a1L	AD<15:0> (address) valid to ALE↓ (address setup time)	0.25Tcy - 10	—	—	ns	
151	Ta1L2ad1	ALE↓ to address out invalid (address hold time)	0	—	—	ns	
152	TadV2wrL	Data out valid to WR↓ (data setup time)	0.25Tcy - 40	—	—	ns	
153	TwrH2ad1	WR↑ to data out invalid (data hold time)	—	0.25Tcy §	—	ns	
154	TwrL	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.

**FIGURE 19-12: MEMORY INTERFACE READ TIMING (NOT SUPPORTED IN PIC17LC4X DEVICES)**



**TABLE 19-12: MEMORY INTERFACE READ REQUIREMENTS (NOT SUPPORTED IN PIC17LC4X DEVICES)**

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
150	TadV2a1L	AD15:AD0 (address) valid to ALE↓ (address setup time)	0.25Tcy - 10	—	—	ns	
151	Ta1L2ad1	ALE↓ to address out invalid (address hold time)	5*	—	—	ns	
160	TadZ2oeL	AD15:AD0 hi-impedance to OE↓	0*	—	—	ns	
161	ToeH2adD	OE↑ to AD15:AD0 driven	0.25Tcy - 15	—	—	ns	
162	TadV2oeH	Data in valid before OE↑ (data setup time)	35	—	—	ns	
163	ToeH2ad1	OE↑ to data in invalid (data hold time)	0	—	—	ns	
164	Ta1H	ALE pulse width	—	0.25Tcy §	—	ns	
165	ToeL	OE pulse width	0.5Tcy - 35 §	—	—	ns	
166	Ta1H2a1H	ALE↑ to ALE↑ (cycle time)	—	Tcy §	—	ns	
167	Tacc	Address access time	—	—	0.75Tcy - 30	ns	
168	Toe	Output enable access time (OE low to Data Valid)	—	—	0.5Tcy - 45	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.

# PIC17C4X

---

NOTES:



## 20.0 PIC17CR42/42A/43/R43/44 DC AND AC CHARACTERISTICS

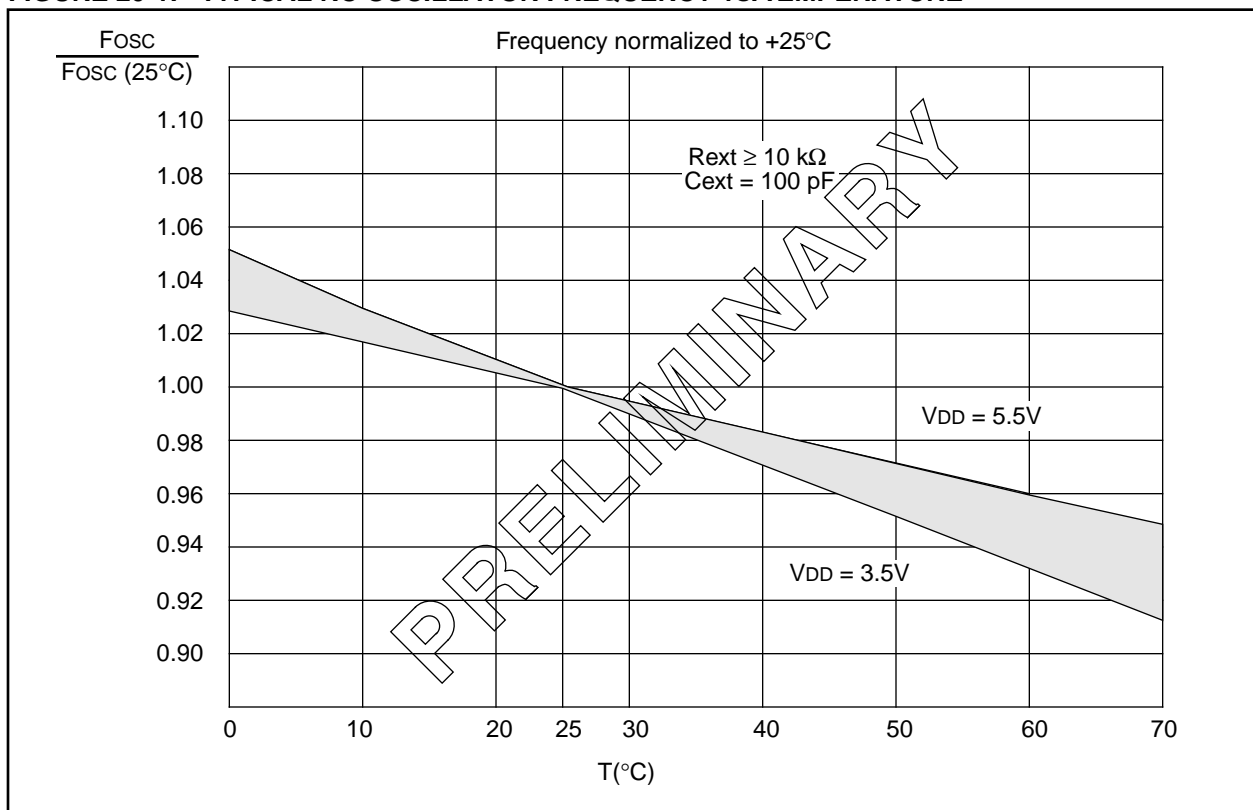
The graphs and tables provided in this section are for design guidance and are not tested nor guaranteed. In some graphs or tables the data presented is outside specified operating range (e.g. outside specified  $V_{DD}$  range). This is for information only and devices are ensured to operate properly only within the specified range.

The data presented in this section is a statistical summary of data collected on units from different lots over a period of time. "Typical" represents the mean of the distribution while "max" or "min" represents  $(\text{mean} + 3\sigma)$  and  $(\text{mean} - 3\sigma)$  respectively where  $\sigma$  is standard deviation.

**TABLE 20-1: PIN CAPACITANCE PER PACKAGE TYPE**

Pin Name	Typical Capacitance (pF)			
	40-pin DIP	44-pin PLCC	44-pin MQFP	44-pin TQFP
All pins, except $\overline{\text{MCLR}}$ , $V_{DD}$ , and $V_{SS}$	10	10	10	10
$\overline{\text{MCLR}}$ pin	20	20	20	20

**FIGURE 20-1: TYPICAL RC OSCILLATOR FREQUENCY vs. TEMPERATURE**



# 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



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



**TABLE 20-2: RC OSCILLATOR FREQUENCIES**

Cext	Rext	Average Fosc @ 5V, 25°C	
		Fosc (MHz)	Tolerance
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%

# PIC17C4X

Applicable Devices 42 R42 42A 43 R43 44

FIGURE 20-5: TRANSCONDUCTANCE (gm) OF LF OSCILLATOR vs. VDD

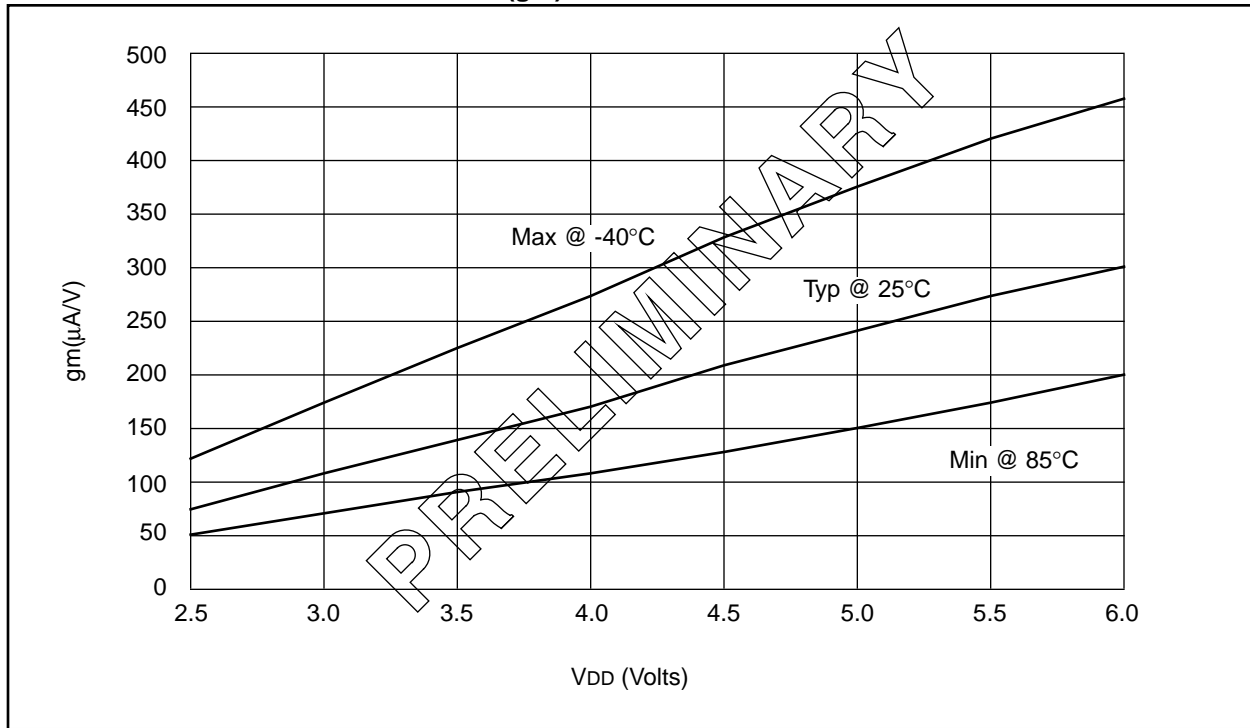
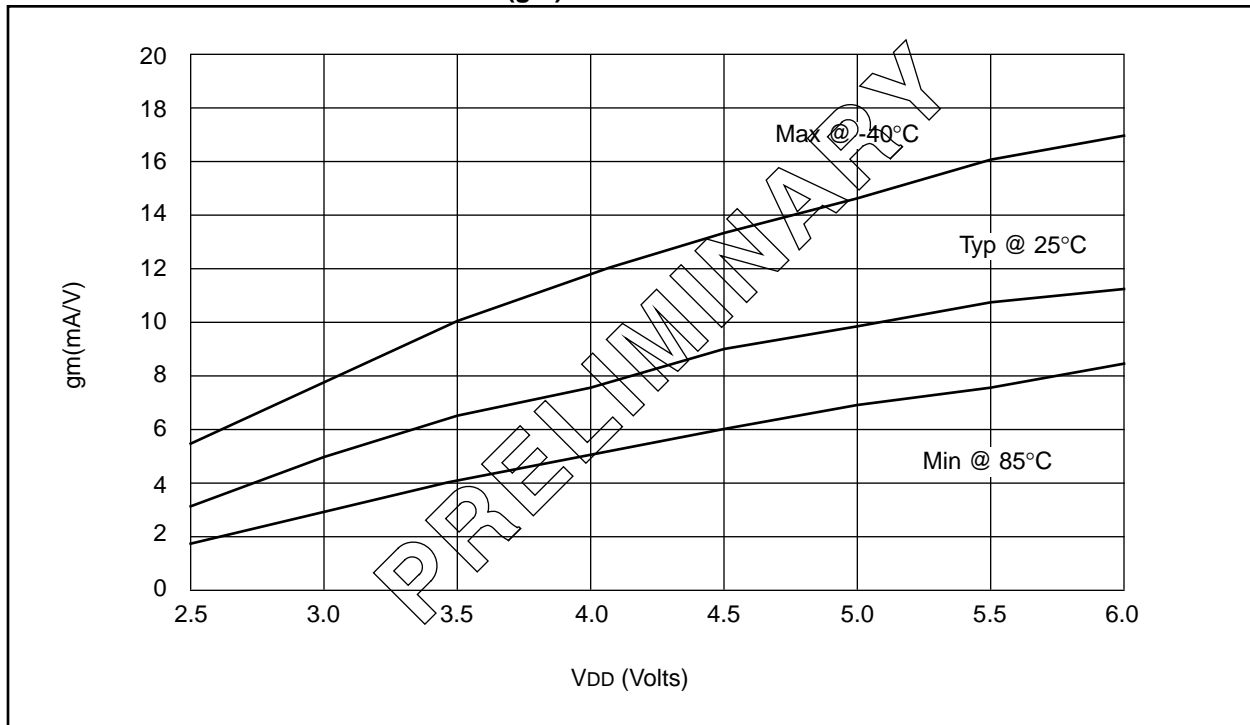


FIGURE 20-6: TRANSCONDUCTANCE (gm) OF XT OSCILLATOR vs. VDD



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



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



# PIC17C4X

Applicable Devices 42 R42 42A 43 R43 44

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



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



**FIGURE 20-11: TYPICAL  $I_{PD}$  vs.  $V_{DD}$  WATCHDOG ENABLED 25°C**



**FIGURE 20-12: MAXIMUM  $I_{PD}$  vs.  $V_{DD}$  WATCHDOG ENABLED**



# PIC17C4X

Applicable Devices 42 R42 42A 43 R43 44

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



FIGURE 20-14: I<sub>OH</sub> vs. V<sub>OH</sub>, VDD = 3V





FIGURE 20-15:  $I_{OH}$  vs.  $V_{OH}$ ,  $V_{DD} = 5V$



FIGURE 20-16:  $I_{OL}$  vs.  $V_{OL}$ ,  $V_{DD} = 3V$



# PIC17C4X

Applicable Devices 42 R42 42A 43 R43 44

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



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



**FIGURE 20-19:  $V_{IH}$ ,  $V_{IL}$  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.  $V_{DD}$**



# PIC17C4X

---

NOTES:

## 21.0 PACKAGING INFORMATION

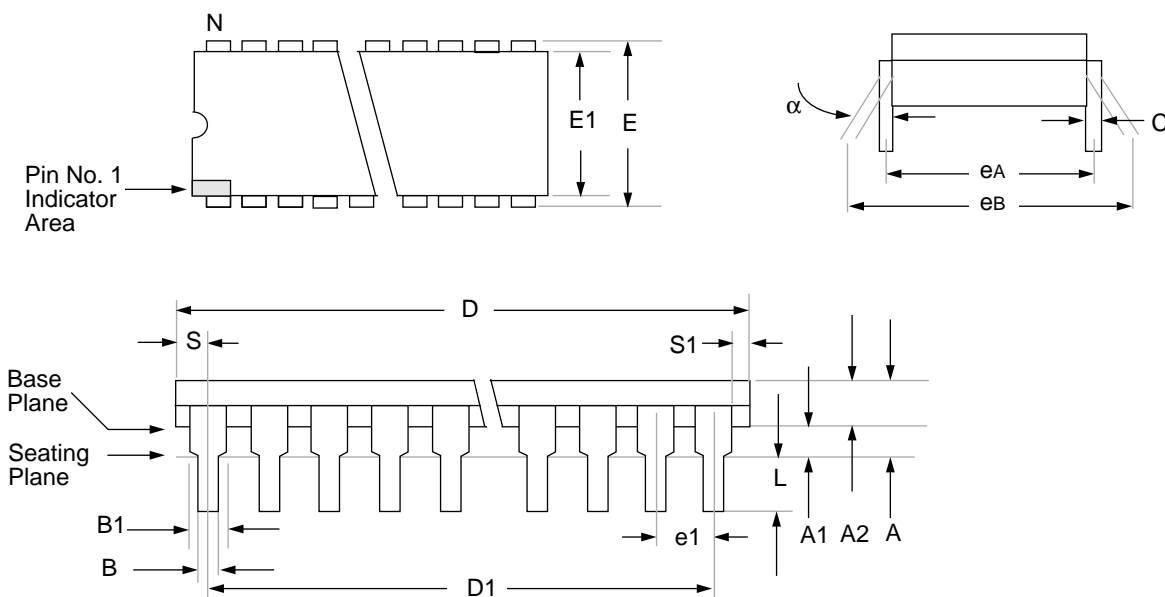
### 21.1 40-Lead Ceramic CERDIP Dual In-line, and CERDIP Dual In-line with Window (600 mil)



Package Group: Ceramic CERDIP Dual In-Line (CDP)						
Symbol	Millimeters			Inches		
	Min	Max	Notes	Min	Max	Notes
$\alpha$	0°	10°		0°	10°	
A	4.318	5.715		0.170	0.225	
A1	0.381	1.778		0.015	0.070	
A2	3.810	4.699		0.150	0.185	
A3	3.810	4.445		0.150	0.175	
B	0.355	0.585		0.014	0.023	
B1	1.270	1.651	Typical	0.050	0.065	Typical
C	0.203	0.381	Typical	0.008	0.015	Typical
D	51.435	52.705		2.025	2.075	
D1	48.260	48.260	Reference	1.900	1.900	Reference
E	15.240	15.875		0.600	0.625	
E1	12.954	15.240		0.510	0.600	
e1	2.540	2.540	Reference	0.100	0.100	Reference
eA	14.986	16.002	Typical	0.590	0.630	Typical
eB	15.240	18.034		0.600	0.710	
L	3.175	3.810		0.125	0.150	
N	40	40		40	40	
S	1.016	2.286		0.040	0.090	
S1	0.381	1.778		0.015	0.070	

# PIC17C4X

## 21.2 40-Lead Plastic Dual In-line (600 mil)



Package Group: Plastic Dual In-Line (PLA)						
Symbol	Millimeters			Inches		
	Min	Max	Notes	Min	Max	Notes
$\alpha$	0°	10°		0°	10°	
A	—	5.080		—	0.200	
A1	0.381	—		0.015	—	
A2	3.175	4.064		0.125	0.160	
B	0.355	0.559		0.014	0.022	
B1	1.270	1.778	Typical	0.050	0.070	Typical
C	0.203	0.381	Typical	0.008	0.015	Typical
D	51.181	52.197		2.015	2.055	
D1	48.260	48.260	Reference	1.900	1.900	Reference
E	15.240	15.875		0.600	0.625	
E1	13.462	13.970		0.530	0.550	
e1	2.489	2.591	Typical	0.098	0.102	Typical
eA	15.240	15.240	Reference	0.600	0.600	Reference
eB	15.240	17.272		0.600	0.680	
L	2.921	3.683		0.115	0.145	
N	40	40		40	40	
S	1.270	—		0.050	—	
S1	0.508	—		0.020	—	

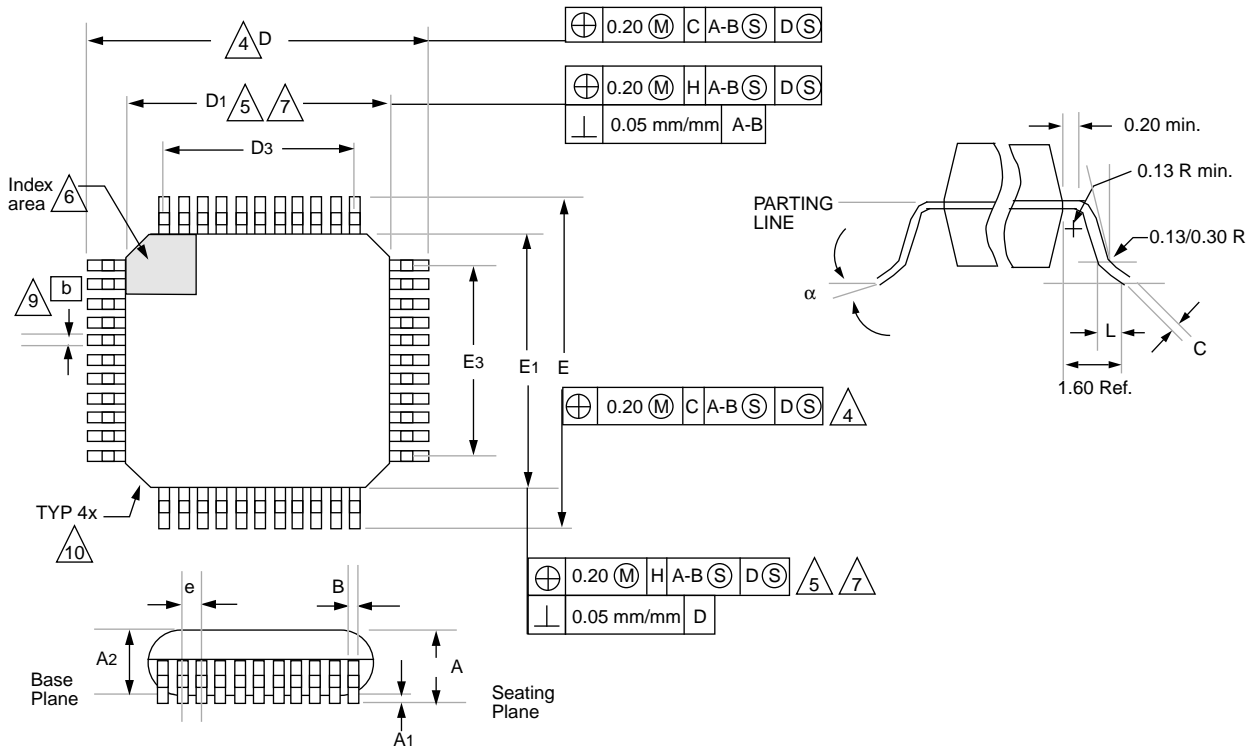
## 21.3 44-Lead Plastic Leaded Chip Carrier (Square)



Package Group: Plastic Leaded Chip Carrier (PLCC)						
Symbol	Millimeters			Inches		
	Min	Max	Notes	Min	Max	Notes
A	4.191	4.572		0.165	0.180	
A1	2.413	2.921		0.095	0.115	
D	17.399	17.653		0.685	0.695	
D1	16.510	16.663		0.650	0.656	
D2	15.494	16.002		0.610	0.630	
D3	12.700	12.700	Reference	0.500	0.500	Reference
E	17.399	17.653		0.685	0.695	
E1	16.510	16.663		0.650	0.656	
E2	15.494	16.002		0.610	0.630	
E3	12.700	12.700	Reference	0.500	0.500	Reference
N	44	44		44	44	
CP	—	0.102		—	0.004	
LT	0.203	0.381		0.008	0.015	

# PIC17C4X

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



Package Group: Plastic MQFP						
Symbol	Millimeters			Inches		
	Min	Max	Notes	Min	Max	Notes
$\alpha$	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	—	



## 21.5 44-Lead Plastic Surface Mount (TQFP 10x10 mm Body 1.0/0.10 mm Lead Form)



Package Group: Plastic TQFP						
Symbol	Millimeters			Inches		
	Min	Max	Notes	Min	Max	Notes
A	1.00	1.20		0.039	0.047	
A1	0.05	0.15		0.002	0.006	
A2	0.95	1.05		0.037	0.041	
D	11.75	12.25		0.463	0.482	
D1	9.90	10.10		0.390	0.398	
E	11.75	12.25		0.463	0.482	
E1	9.90	10.10		0.390	0.398	
L	0.45	0.75		0.018	0.030	
e	0.80 BSC			0.031 BSC		
b	0.30	0.45		0.012	0.018	
b1	0.30	0.40		0.012	0.016	
c	0.09	0.20		0.004	0.008	
c1	0.09	0.16		0.004	0.006	
N	44	44		44	44	
$\theta$	$0^\circ$	$7^\circ$		$0^\circ$	$7^\circ$	

Note 1: Dimensions D1 and E1 do not include mold protrusion. Allowable mold protrusion is 0.25m/m (0.010") per side. D1 and E1 dimensions including mold mismatch.

2: Dimension "b" does not include Dambar protrusion, allowable Dambar protrusion shall be 0.08m/m (0.003")max.

3: This outline conforms to JEDEC MS-026.

# PIC17C4X

## 21.6 Package Marking Information

### 40-Lead PDIP/CERDIP



### Example



### 40 Lead CERDIP Windowed



### Example



### 44-Lead PLCC



### Example



### 44-Lead MQFP



### Example



### 44-Lead TQFP



### Example



<b>Legend:</b> MM...M	Microchip part number information
XX...X	Customer specific information*
AA	Year code (last 2 digits of calendar year)
BB	Week code (week of January 1 is week '01')
C	Facility code of the plant at which wafer is manufactured
	C = Chandler, Arizona, U.S.A.,
	S = Tempe, Arizona, U.S.A.
D	Mask revision number
E	Assembly code of the plant or country of origin in which part was assembled

**Note:** In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line thus limiting the number of available characters for customer specific information.

\* Standard OTP marking consists of Microchip part number, year code, week code, facility code, mask rev#, and assembly code. For OTP marking beyond this, certain price adders apply. Please check with your Microchip Sales Office. For QTP devices, any special marking adders are included in QTP price.

## APPENDIX A: MODIFICATIONS

The following is the list of modifications over the PIC16CXX microcontroller family:

1. Instruction word length is increased to 16-bit. This allows larger page sizes both in program memory (8 Kwords versus 2 Kwords) and register file (256 bytes versus 128 bytes).
2. Four modes of operation: microcontroller, protected microcontroller, extended microcontroller, and microprocessor.
3. 22 new instructions. The `MOVF`, `TRIS` and `OPTION` instructions have been removed.
4. 4 new instructions for transferring data between data memory and program memory. This can be used to "self program" the EPROM program memory.
5. Single cycle data memory to data memory transfers possible (`MOVFP` and `MOVFP` instructions). These instructions do not affect the Working register (`WREG`).
6. `W` register (`WREG`) is now directly addressable.
7. A PC high latch register (`PCLATH`) is extended to 8-bits. The `PCLATCH` register is now both readable and writable.
8. Data memory paging is redefined slightly.
9. `DDR` registers replaces function of `TRIS` registers.
10. Multiple Interrupt vectors added. This can decrease the latency for servicing the interrupt.
11. Stack size is increased to 16 deep.
12. `BSR` register for data memory paging.
13. Wake up from `SLEEP` operates slightly differently.
14. The Oscillator Start-Up Timer (`OST`) and Power-Up Timer (`PWRT`) operate in parallel and not in series.
15. `PORTB` interrupt on change feature works on all eight port pins.
16. `TMR0` is 16-bit plus 8-bit prescaler.
17. Second indirect addressing register added (`FSR1` and `FSR2`). Configuration bits can select the `FSR` registers to auto-increment, auto-decrement, remain unchanged after an indirect address.
18. Hardware multiplier added ( $8 \times 8 \rightarrow 16$ -bit) (`PIC17C43` and `PIC17C44` only).
19. Peripheral modules operate slightly differently.
20. Oscillator modes slightly redefined.
21. Control/Status bits and registers have been placed in different registers and the control bit for globally enabling interrupts has inverse polarity.
22. Addition of a test mode pin.
23. In-circuit serial programming is not implemented.

## APPENDIX B: COMPATIBILITY

To convert code written for `PIC16CXX` to `PIC17CXX`, the user should take the following steps:

1. Remove any `TRIS` and `OPTION` instructions, and implement the equivalent code.
2. Separate the interrupt service routine into its four vectors.
3. Replace:
 

```
MOVF    REG1, W
with:
MOVFP   REG1, WREG
```
4. Replace:
 

```
MOVF    REG1, W
MOVWF   REG2
with:
MOVFP   REG1, REG2 ; Addr(REG1) < 20h
or
MOVFP   REG1, REG2 ; Addr(REG2) < 20h
```

**Note:** If `REG1` and `REG2` are both at addresses greater than `20h`, two instructions are required.

```
MOVFP   REG1, WREG ;
MOVFP   WREG, REG2 ;
```

5. Ensure that all bit names and register names are updated to new data memory map location.
6. Verify data memory banking.
7. Verify mode of operation for indirect addressing.
8. Verify peripheral routines for compatibility.
9. Weak pull-ups are enabled on reset.

To convert code from the `PIC17C42` to all the other `PIC17C4X` devices, the user should take the following steps.

1. If the hardware multiply is to be used, ensure that any variables at address `18h` and `19h` are moved to another address.
2. Ensure that the upper nibble of the `BSR` was not written with a non-zero value. This may cause unexpected operation since the `RAM` bank is no longer `0`.
3. The disabling of global interrupts has been enhanced so there is no additional testing of the `GLINTD` bit after a `BSF CPUSTA, GLINTD` instruction.

## APPENDIX C: WHAT'S NEW

The structure of the document has been made consistent with other data sheets. This ensures that important topics are covered across all PIC16/17 families. Here is an overview of new features.

Added the following devices:

PIC17CR42

PIC17C42A

PIC17CR43

A 33 MHz option is now available.

## APPENDIX D: WHAT'S CHANGED

To make software more portable across the different PIC16/17 families, the name of several registers and control bits have been changed. This allows control bits that have the same function, to have the same name (regardless of processor family). Care must still be taken, since they may not be at the same special function register address. The following shows the register and bit names that have been changed:

Old Name	New Name
TX8/9	TX9
RC8/9	RX9
RCD8	RX9D
TXD8	TX9D

Instruction DECFSNZ corrected to DCFSNZ

Instruction INCFSNZ corrected to INFSNZ

Enhanced discussion on PWM to include equation for determining bits of PWM resolution.

Section 13.2.2 and 13.3.2 have had the description of updating the FERR and RX9 bits enhanced.

The location of configuration bit PM2 was changed (Figure 6-1 and Figure 14-1).

Enhanced description of the operation of the INTSTA register.

Added note to discussion of interrupt operation.

Tightened electrical spec D110.

Corrected steps for setting up USART Asynchronous Reception.



# PIC17C4X

## E.2 PIC16C5X Family of Devices

Device	Clock		Memory		Peripherals		Features		
	Maximum Frequency of Operation (MHz)	Program Memory (x12 words)	RAM Data Memory (bytes)	Timer Module(s)	I/O Pins	Voltage Range (Volts)	Number of Instructions	Packages	
PIC16C52	4	384	—	25	TMR0	12	2.5-6.25	33	18-pin DIP, SOIC
PIC16C54	20	512	—	25	TMR0	12	2.5-6.25	33	18-pin DIP, SOIC; 20-pin SSOP
PIC16C54A	20	512	—	25	TMR0	12	2.0-6.25	33	18-pin DIP, SOIC; 20-pin SSOP
PIC16CR54A	20	—	512	25	TMR0	12	2.0-6.25	33	18-pin DIP, SOIC; 20-pin SSOP
PIC16C55	20	512	—	24	TMR0	20	2.5-6.25	33	28-pin DIP, SOIC, SSOP
PIC16C56	20	1K	—	25	TMR0	12	2.5-6.25	33	18-pin DIP, SOIC; 20-pin SSOP
PIC16C57	20	2K	—	72	TMR0	20	2.5-6.25	33	28-pin DIP, SOIC, SSOP
PIC16CR57B	20	—	2K	72	TMR0	20	2.5-6.25	33	28-pin DIP, SOIC, SSOP
PIC16C58A	20	2K	—	73	TMR0	12	2.0-6.25	33	18-pin DIP, SOIC; 20-pin SSOP
PIC16CR58A	20	—	2K	73	TMR0	12	2.5-6.25	33	18-pin DIP, SOIC; 20-pin SSOP

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

## E.3 PIC16CXXX Family of Devices

Device	Clock		Memory		Peripherals			Features		
	Maximum Frequency of Operation (MHz)	Program Memory (K14 words)	Data Memory (bytes)	Timer Module(s)	Comparators	Internal Reference Voltage	Interrupt Sources	I/O Pins	Voltage Range (Volts)	Brown-out Reset Packages
PIC16C554	20	512	80	TMR0	—	3	13	2.5-6.0	—	18-pin DIP; SOIC; 20-pin SSOP
PIC16C556	20	1K	80	TMR0	—	3	13	2.5-6.0	—	18-pin DIP; SOIC; 20-pin SSOP
PIC16C558	20	2K	128	TMR0	—	3	13	2.5-6.0	—	18-pin DIP; SOIC; 20-pin SSOP
PIC16C620	20	512	80	TMR0	2	Yes	4	2.5-6.0	Yes	18-pin DIP; SOIC; 20-pin SSOP
PIC16C621	20	1K	80	TMR0	2	Yes	4	2.5-6.0	Yes	18-pin DIP; SOIC; 20-pin SSOP
PIC16C622	20	2K	128	TMR0	2	Yes	4	2.5-6.0	Yes	18-pin DIP; SOIC; 20-pin SSOP

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

All PIC16C6XXX Family devices use serial programming with clock pin RB6 and data pin RB7.

# PIC17C4X

## E.4 PIC16C6X Family of Devices

Device	Clock		Memory				Peripherals				Features			
	Program Memory (Kx14 words)	Maximum Frequency of Operation (MHz)	ROM	EPROM	Data Memory (bytes)	Timer Module(s)	Serial Ports (SPI/I <sup>2</sup> C, USART)	Parallel Slave Port	I/O Pins	Voltage Range (Volts)	In-Circuit Serial Programming	Brown-out Reset	Packages	
PIC16C62	20	2K	—	—	128	TMR0, TMR1, TMR2	1 SPI/I <sup>2</sup> C	—	7	22	3.0-6.0	Yes	—	28-pin SDIP, SOIC, SSOP
PIC16C62A <sup>(1)</sup>	20	2K	—	—	128	TMR0, TMR1, TMR2	1 SPI/I <sup>2</sup> C	—	7	22	2.5-6.0	Yes	Yes	28-pin SDIP, SOIC, SSOP
PIC16CR62 <sup>(1)</sup>	20	—	2K	—	128	TMR0, TMR1, TMR2	1 SPI/I <sup>2</sup> C	—	7	22	2.5-6.0	Yes	Yes	28-pin SDIP, SOIC, SSOP
PIC16C63	20	4K	—	—	192	TMR0, TMR1, TMR2	2 SPI/I <sup>2</sup> C, USART	—	10	22	2.5-6.0	Yes	Yes	28-pin SDIP, SOIC
PIC16CR63 <sup>(1)</sup>	20	—	4K	—	192	TMR0, TMR1, TMR2	2 SPI/I <sup>2</sup> C, USART	—	10	22	2.5-6.0	Yes	Yes	28-pin SDIP, SOIC
PIC16C64	20	2K	—	—	128	TMR0, TMR1, TMR2	1 SPI/I <sup>2</sup> C	Yes	8	33	3.0-6.0	Yes	—	40-pin DIP; 44-pin PLCC, MQFP
PIC16C64A <sup>(1)</sup>	20	2K	—	—	128	TMR0, TMR1, TMR2	1 SPI/I <sup>2</sup> C	Yes	8	33	2.5-6.0	Yes	Yes	40-pin DIP; 44-pin PLCC, MQFP, TQFP
PIC16CR64 <sup>(1)</sup>	20	—	2K	—	128	TMR0, TMR1, TMR2	1 SPI/I <sup>2</sup> C	Yes	8	33	2.5-6.0	Yes	Yes	40-pin DIP; 44-pin PLCC, MQFP, TQFP
PIC16C65	20	4K	—	—	192	TMR0, TMR1, TMR2	2 SPI/I <sup>2</sup> C, USART	Yes	11	33	3.0-6.0	Yes	—	40-pin DIP; 44-pin PLCC, MQFP
PIC16C65A <sup>(1)</sup>	20	4K	—	—	192	TMR0, TMR1, TMR2	2 SPI/I <sup>2</sup> C, USART	Yes	11	33	2.5-6.0	Yes	Yes	40-pin DIP; 44-pin PLCC, MQFP, TQFP
PIC16CR65 <sup>(1)</sup>	20	—	4K	—	192	TMR0, TMR1, TMR2	2 SPI/I <sup>2</sup> C, USART	Yes	11	33	2.5-6.0	Yes	Yes	40-pin DIP; 44-pin PLCC, MQFP, TQFP

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

All PIC16C6X family devices use serial programming with clock pin RB6 and data pin RB7.

Note 1: Please contact your local sales office for availability of these devices.



## E.5 PIC16C7X Family of Devices

Device	Clock		Memory		Peripherals				Features			
	Maximum Frequency of Operation (MHz)	EPROM Program Memory (K x 4 words)	Data Memory (bytes)	Timer Modules(s)	Serial Ports (SPI/I <sup>2</sup> C, USART)	A/D Converter (8-bit) Channels	Interrupt Sources	I/O Pins	Voltage Range (Volts)	In-Circuit Serial Programming	Brown-out Reset	Packages
PIC16C710	20	512	36	TMR0	—	—	4	4	13	3.0-6.0	Yes	18-pin DIP, SOIC; 20-pin SSOP
PIC16C71	20	1K	36	TMR0	—	—	4	4	13	3.0-6.0	Yes	18-pin DIP, SOIC
PIC16C711	20	1K	68	TMR0	—	—	4	4	13	3.0-6.0	Yes	18-pin DIP, SOIC; 20-pin SSOP
PIC16C72	20	2K	128	TMR0, TMR1, TMR2	1	SPI/I <sup>2</sup> C	—	5	8	2.5-6.0	Yes	28-pin SDIP, SOIC, SSOP
PIC16C73	20	4K	192	TMR0, TMR1, TMR2	2	SPI/I <sup>2</sup> C, USART	—	5	11	3.0-6.0	Yes	28-pin SDIP, SOIC
PIC16C73A <sup>(1)</sup>	20	4K	192	TMR0, TMR1, TMR2	2	SPI/I <sup>2</sup> C, USART	—	5	11	2.5-6.0	Yes	28-pin SDIP, SOIC
PIC16C74	20	4K	192	TMR0, TMR1, TMR2	2	SPI/I <sup>2</sup> C, USART	Yes	8	12	3.0-6.0	Yes	40-pin DIP; 44-pin PLCC, MQFP
PIC16C74A <sup>(1)</sup>	20	4K	192	TMR0, TMR1, TMR2	2	SPI/I <sup>2</sup> C, USART	Yes	8	12	2.5-6.0	Yes	40-pin DIP; 44-pin PLCC, MQFP, TQFP

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

All PIC16C7X Family devices use serial programming with clock pin RB6 and data pin RB7.

Note 1: Please contact your local sales office for availability of these devices.

# PIC17C4X

## E.6 PIC16C8X Family of Devices

	Clock		Memory			Peripherals		Features			
	Maximum Frequency of Operation (MHz)	Program Memory	Data Memory (bytes)	Data EEPROM (bytes)	Timer Modules	Interrupt Sources	I/O Pins				
PIC16C84	10	—	1K	—	36	64	TMR0	4	13	2.0-6.0	18-pin DIP, SOIC
PIC16F84 <sup>(1)</sup>	10	1K	—	—	68	64	TMR0	4	13	2.0-6.0	18-pin DIP, SOIC
PIC16CR84 <sup>(1)</sup>	10	—	—	1K	68	64	TMR0	4	13	2.0-6.0	18-pin DIP, SOIC
PIC16F83 <sup>(1)</sup>	10	512	—	—	36	64	TMR0	4	13	2.0-6.0	18-pin DIP, SOIC
PIC16CR83 <sup>(1)</sup>	10	—	—	512	36	64	TMR0	4	13	2.0-6.0	18-pin DIP, SOIC

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

All PIC16C8X family devices use serial programming with clock pin RB6 and data pin RB7.

Note 1: Please contact your local sales office for availability of these devices.

## E.7 PIC16C9XX Family Of Devices

Device	Clock		Memory		Peripherals					Features					
	Maximum Frequency of Operation (MHz)	Program Memory (Kbytes)	Data Memory (bytes)	Timer Module(s)	EPRAM	Serial Port(s) (SPI/I <sup>2</sup> C, USART)	Parallel Slave Port	A/D Converter (8-bit) Channels	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/I <sup>2</sup> C	—	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/I <sup>2</sup> C	—	5	4 Com 32 Seg	9	25	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.

# PIC17C4X

## E.8 PIC17CXX Family of Devices

Device	Clock			Memory			Peripherals					Features		
	Maximum Frequency of Operation (MHz)	Program Memory (Words)	RAM Data Memory (bytes)	Timer Module(s)	Captures	PMs	Serial Port(s) (USART)	Hardware Multiply	External Interrupts	Interrupt Sources	I/O Pins	Voltage Range (Volts)	Number of Instructions	Packages
PIC17C42	25	2K	—	232	2	2	Yes	—	Yes	11	33	4.5-5.5	55	40-pin DIP; 44-pin PLCC, MQFP
PIC17C42A	25	2K	—	232	2	2	Yes	Yes	Yes	11	33	2.5-6.0	58	40-pin DIP; 44-pin PLCC, TQFP, MQFP
PIC17CR42	25	—	2K	232	2	2	Yes	Yes	Yes	11	33	2.5-6.0	58	40-pin DIP; 44-pin PLCC, TQFP, MQFP
PIC17C43	25	4K	—	454	2	2	Yes	Yes	Yes	11	33	2.5-6.0	58	40-pin DIP; 44-pin PLCC, TQFP, MQFP
PIC17CR43	25	—	4K	454	2	2	Yes	Yes	Yes	11	33	2.5-6.0	58	40-pin DIP; 44-pin PLCC, TQFP, MQFP
PIC17C44	25	8K	—	454	2	2	Yes	Yes	Yes	11	33	2.5-6.0	58	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.

## PIN COMPATIBILITY

Devices that have the same package type and VDD, VSS and MCLR pin locations are said to be pin compatible. This allows these different devices to operate in the same socket. Compatible devices may only require minor software modification to allow proper operation in the application socket (ex., PIC16C56 and PIC16C61 devices). Not all devices in the same package size are pin compatible; for example, the PIC16C62 is compatible with the PIC16C63, but not the PIC16C55.

Pin compatibility does not mean that the devices offer the same features. As an example, the PIC16C54 is pin compatible with the PIC16C71, but does not have an A/D converter, weak pull-ups on PORTB, or interrupts.

**TABLE E-1: PIN COMPATIBLE DEVICES**

Pin Compatible Devices	Package
PIC12C508, PIC12C509	8-pin
PIC16C54, PIC16C54A, PIC16CR54A, PIC16C56, PIC16C58A, PIC16CR58A, PIC16C61, PIC16C554, PIC16C556, PIC16C558 PIC16C620, PIC16C621, PIC16C622, PIC16C710, PIC16C71, PIC16C711, PIC16F83, PIC16CR83, PIC16C84, PIC16F84A, PIC16CR84	18-pin 20-pin
PIC16C55, PIC16C57, PIC16CR57B	28-pin
PIC16C62, PIC16CR62, PIC16C62A, PIC16C63, PIC16C72, PIC16C73, PIC16C73A	28-pin
PIC16C64, PIC16CR64, PIC16C64A, PIC16C65, PIC16C65A, PIC16C74, PIC16C74A	40-pin
PIC17C42, PIC17CR42, PIC17C42A, PIC17C43, PIC17CR43, PIC17C44	40-pin
PIC16C923, PIC16C924	64/68-pin

# PIC17C4X

---

NOTES:

## APPENDIX F: ERRATA FOR PIC17C42 SILICON

The PIC17C42 devices that you have received have the following anomalies. At present there is no intention for future revisions to the present PIC17C42 silicon. If these cause issues for the application, it is recommended that you select the PIC17C42A device.

**Note:** New designs should use the PIC17C42A.

1. When the Oscillator Start-Up Timer (OST) is enabled (in LF or XT oscillator modes), any interrupt that wakes the processor may cause a WDT reset. This occurs when the WDT is greater than or equal to 50% time-out period when the `SLEEP` instruction is executed. This will not occur in either the EC or RC oscillator modes.

### Work-arounds

- a) Always ensure that the `CLRWDT` instruction is executed before the WDT increments past 50% of the WDT period. This will keep the "false" WDT reset from occurring.
- b) When using the WDT as a normal timer (WDT disabled), ensure that the WDT is less than or equal to 50% time-out period when the `SLEEP` instruction is executed. This can be done by monitoring the  $\overline{TO}$  bit for changing state from set to clear. Example 1 shows putting the PIC17C42 to sleep.

### EXAMPLE F-1: PIC17C42 TO SLEEP

```

BTFFS    CPUSTA, TO    ; TO = 0?
CLRWDT
LOOP BTFFS    CPUSTA, TO    ; WDT rollover?
GOTO     LOOP          ; NO, wait
SLEEP

```

2. When the clock source of Timer1 or Timer2 is selected to external clock, the overflow interrupt flag will be set twice, once when the timer equals the period, and again when the timer value is reset to 0h. If the latency to clear `TMRxIF` is greater than the time to the next clock pulse, no problems will be noticed. If the latency is less than the time to the next timer clock pulse, the interrupt will be serviced twice.

### Work-arounds

- a) Ensure that the timer has rolled over to 0h before clearing the flag bit.
- b) Clear the timer in software. Clearing the timer in software causes the period to be one count less than expected.

## Design considerations

The device must not be operated outside of the specified voltage range. An external reset circuit must be used to ensure the device is in reset when a brown-out occurs or the `VDD` rise time is too long. Failure to ensure that the device is in reset when device voltage is out of specification may cause the device to lock-up and ignore the `MCLR` pin.

# PIC17C4X

---

NOTES:



## 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
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/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

# PIC17C4X

Delay From External Clock Edge .....	68
Development Support .....	143
Development Tools .....	143
Device Drawings	
44-Lead Plastic Surface Mount (MQFP	
10x10 mm Body 1.6/0.15 mm Lead Form) .....	209
DIGIT BORROW .....	9
Digit Carry (DC) .....	9
Duty Cycle .....	75

## E

Electrical Characteristics	
PIC17C42	
Absolute Maximum Ratings .....	147
Capture Timing .....	159
CLKOUT and I/O Timing .....	156
DC Characteristics .....	149
External Clock Timing .....	155
Memory Interface Read Timing .....	162
Memory Interface Write Timing .....	161
PWM Timing .....	159
RESET, Watchdog Timer, Oscillator Start-up	
Timer and Power-up Timer .....	157
Timer0 Clock Timings .....	158
Timer1, Timer2 and Timer3 Clock Timing .....	158
USART Module, Synchronous Receive .....	160
USART Module, Synchronous Transmission .....	160
PIC17C43/44	
Absolute Maximum Ratings .....	175
Capture Timing .....	188
CLKOUT and I/O Timing .....	185
DC Characteristics .....	177
External Clock Timing .....	184
Memory Interface Read Timing .....	191
Memory Interface Write Timing .....	190
Parameter Measurement Information .....	183
RESET, Watchdog Timer, Oscillator Start-up	
Timer and Power-up Timer Timing .....	186
Timer0 Clock Timing .....	187
Timer1, Timer2 and Timer3 Clock Timing .....	187
Timing Parameter Symbolology .....	182
USART Module Synchronous Receive	
Timing .....	189
USART Module Synchronous Transmission	
Timing .....	189
EPROM Memory Access Time Order Suffix .....	31
Extended Microcontroller .....	29
Extended Microcontroller Mode .....	31
External Memory Interface .....	31
External Program Memory Waveforms .....	31

## F

Family of Devices .....	6
PIC14000 .....	213
PIC16C5X .....	214
PIC16CXXX .....	215
PIC16C6X .....	216
PIC16C7X .....	217
PIC16C8X .....	218
PIC16C9XX .....	219
PIC17CXX .....	220
FERR .....	84, 91
FOSC0 .....	99

FOSC1 .....	99
FS0 .....	36
FS1 .....	36
FS2 .....	36
FS3 .....	36
FSR0 .....	34, 40
FSR1 .....	34, 40
Fuzzy Logic Dev. System ( <i>fuzzyTECH</i> <sup>®</sup> -MP) .....	143, 145

## G

General Format for Instructions .....	108
General Purpose RAM .....	29
General Purpose RAM Bank .....	42
General Purpose Register (GPR) .....	32
GLINTD .....	25, 37, 78, 105
GOTO .....	122
GPR (General Purpose Register) .....	32
Graphs	
IOH vs. VOH, VDD = 3V .....	170, 200
IOH vs. VOH, VDD = 5V .....	171, 201
IOL vs. VOL, VDD = 3V .....	171, 201
IOL vs. VOL, VDD = 5V .....	172, 202
Maximum IDD vs. Frequency	
(External Clock 125°C to -40°C) .....	167, 197
Maximum IPD vs. VDD Watchdog Disabled .....	168, 198
Maximum IPD vs. VDD Watchdog Enabled .....	169, 199
RC Oscillator Frequency vs.	
VDD (Cext = 100 pF) .....	164, 194
RC Oscillator Frequency vs.	
VDD (Cext = 22 pF) .....	164, 194
RC Oscillator Frequency vs.	
VDD (Cext = 300 pF) .....	165, 195
Transconductance of LF Oscillator vs. VDD .....	166, 196
Transconductance of XT Oscillator vs. VDD .....	166, 196
Typical IDD vs. Frequency	
(External Clock 25°C) .....	167, 197
Typical IPD vs. VDD Watchdog Disabled 25°C ..	168, 198
Typical IPD vs. VDD Watchdog Enabled 25°C ..	169, 199
Typical RC Oscillator vs. Temperature .....	163, 193
VTH (Input Threshold Voltage) of I/O Pins vs.	
VDD .....	172, 202
VTH (Input Threshold Voltage) of OSC1 Input	
(In XT, HS, and LP Modes) vs. VDD .....	173, 203
VTH, VIL of $\overline{\text{MCLR}}$ , T0CKI and OSC1	
(In RC Mode) vs. VDD .....	173, 203
WDT Timer Time-Out Period vs. VDD .....	170, 200

## H

Hardware Multiplier .....	49
---------------------------	----

## I

I/O Ports	
Bi-directional .....	64
I/O Ports .....	53
Programming Considerations .....	64
Read-Modify-Write Instructions .....	64
Successive Operations .....	64
INCF .....	123
INCFSNZ .....	124
INCFSZ .....	123
INDFO .....	34, 40
INDF1 .....	34, 40

Indirect Addressing		TSTFSZ	140
Indirect Addressing	39	XORLW	141
Operation	40	XORWF	141
Registers	40	Instruction Set Summary	107
Initialization Conditions For Special Function Registers	19	INT Pin	26
Initializing PORTB	57	INTE	22
Initializing PORTC	58	INTEDG	38, 67
Initializing PORTD	60	Interrupt on Change Feature	55
Initializing PORTE	62	Interrupt Status Register (INTSTA)	22
Instruction Flow/Pipelining	14	Interrupts	
Instruction Set	110	Context Saving	27
ADDLW	112	Flag bits	
ADDWF	112	TMR1IE	21
ADDWFC	113	TMR1IF	21
ANDLW	113	TMR2IE	21
ANDWF	114	TMR2IF	21
BCF	114	TMR3IE	21
BSF	115	TMR3IF	21
BTFSC	115	Interrupts	21
BTFSS	116	Logic	21
BTG	116	Operation	25
CALL	117	Peripheral Interrupt Enable	23
CLRF	117	Peripheral Interrupt Request	24
CLRWDT	118	PWM	76
COMF	118	Status Register	22
CPFSEQ	119	Table Write Interaction	45
CPFSGT	119	Timing	26
CPFSLT	120	Vectors	
DAW	120	Peripheral Interrupt	26
DECF	121	RA0/INT Interrupt	26
DECFSNZ	122	T0CKI Interrupt	26
DECFSZ	121	TMR0 Interrupt	26
GOTO	122	Vectors/Priorities	25
INCF	123	Wake-up from SLEEP	105
INCFSNZ	124	INTF	22
INCFSZ	123	INTSTA	34
IORLW	124	INTSTA Register	22
IORWF	125	IORLW	124
LCALL	125	IORWF	125
MOVFP	126		
MOVLB	126	<b>L</b>	
MOVLR	127	LCALL	125
MOVLW	127	Long Writes	45
MOVPF	128		
MOVWF	128	<b>M</b>	
MULLW	129	Memory	
MULWF	129	External Interface	31
NEGW	130	External Memory Waveforms	31
NOP	130	Memory Map (Different Modes)	30
RETFIE	131	Mode Memory Access	30
RETLW	131	Organization	29
RETURN	132	Program Memory	29
RLCF	132	Program Memory Map	29
RLNCF	133	Microcontroller	29
RRCF	133	Microprocessor	29
RRNCF	134	Minimizing Current Consumption	106
SETF	134	MOVFP	126
SLEEP	135	MOVLB	126
SUBLW	135	MOVLR	127
SUBWF	136	MOVLW	127
SUBWFB	136	MOVPF	128
SWAPF	137	MOVWF	128
TABLRD	137, 138	MPASM Assembler	143, 144
TABLWT	138, 139		
TLRD	139		
TLWT	140		



Receive Status and Control Register .....	83
Register File Map .....	33
Registers	
ALUSTA .....	27, 36
BRG .....	86
BSR .....	27
CPUSTA .....	37
File Map .....	33
FSR0 .....	40
FSR1 .....	40
INDF0 .....	40
INDF1 .....	40
INTSTA .....	22
PIE .....	23
PIR .....	24
RCSTA .....	84
Special Function Table .....	34
T0STA .....	38, 67
TCON1 .....	71
TCON2 .....	72
TMR1 .....	81
TMR2 .....	81
TMR3 .....	81
TXSTA .....	83
WREG .....	27
Reset	
Section .....	15
Status Bits and Their Significance .....	16
Time-Out in Various Situations .....	16
Time-Out Sequence .....	16
RETFIE .....	131
RETLW .....	131
RETURN .....	132
RLCF .....	132
RLNCF .....	133
RRCF .....	133
RRNCF .....	134
RX Pin Sampling Scheme .....	91
RX9 .....	84
RX9D .....	84

## S

Sampling .....	91
Saving STATUS and WREG in RAM .....	27
SETF .....	134
SFR .....	108
SFR (Special Function Registers) .....	29, 32
SFR As Source/Destination .....	108
Signed Math .....	9
SLEEP .....	99, 105, 135
Software Simulator (MPSIM) .....	145
SPBRG .....	19, 34, 92, 96, 98
Special Features of the CPU .....	99
Special Function Registers .....	29, 32, 34, 108
SPEN .....	84
SREN .....	84
Stack	
Operation .....	39
Pointer .....	39
Stack .....	29
STKAL .....	39
STKAV .....	37
SUBLW .....	135
SUBWF .....	136
SUBWFB .....	136

SWAPF .....	137
SYNC .....	83
Synchronous Master Mode .....	93
Synchronous Master Reception .....	95
Synchronous Master Transmission .....	93
Synchronous Slave Mode .....	97

## T

T0CKI Pin .....	26
T0CKIE .....	22
T0CKIF .....	22
T0CS .....	38, 67
T0IE .....	22
T0IF .....	22
T0SE .....	38, 67
T0STA .....	34, 38
T16 .....	71
Table Latch .....	40
Table Pointer .....	40
Table Read	
Example .....	48
Section .....	43
Table Reads Section .....	48
TABLRD Operation .....	44
Timing .....	48
TLRD .....	48
TLRD Operation .....	44
Table Write	
Code .....	46
Interaction .....	45
Section .....	43
TABLWT Operation .....	43
Terminating Long Writes .....	45
Timing .....	46
TLWT Operation .....	43
To External Memory .....	46
To Internal Memory .....	45
TABLRD .....	44, 137, 138
TABLWT .....	43, 138, 139
TBLATH .....	40
TBLATL .....	40
TBLPTRH .....	34, 40
TBLPTRL .....	34, 40
TCLK12 .....	71
TCLK3 .....	71
TCON1 .....	20, 35
TCON2 .....	20, 35
Terminating Long Writes .....	45
Time-Out Sequence .....	16
Timer Resources .....	65
Timer0 .....	67
Timer1	
16-bit Mode .....	74
Clock Source Select .....	71
On bit .....	72
Section .....	71, 73
Timer2	
16-bit Mode .....	74
Clock Source Select .....	71
On bit .....	72
Section .....	71, 73
Timer3	
Clock Source Select .....	71
On bit .....	72
Section .....	71, 77

# PIC17C4X

Timing Diagrams	
Asynchronous Master Transmission	90
Asynchronous Reception	92
Back to Back Asynchronous Master Transmission	90
Interrupt (INT, TMR0 Pins)	26
PIC17C42 Capture	159
PIC17C42 CLKOUT and I/O	156
PIC17C42 Memory Interface Read	162
PIC17C42 Memory Interface Write	161
PIC17C42 PWM Timing	159
PIC17C42 RESET, Watchdog Timer, Oscillator	
Start-up Timer and Power-up Timer	157
PIC17C42 Timer0 Clock	158
PIC17C42 Timer1, Timer2 and Timer3 Clock	158
PIC17C42 USART Module, Synchronous	
Receive	160
PIC17C42 USART Module, Synchronous	
Transmission	160
PIC17C43/44 Capture Timing	188
PIC17C43/44 CLKOUT and I/O	185
PIC17C43/44 External Clock	184
PIC17C43/44 Memory Interface Read	191
PIC17C43/44 Memory Interface Write	190
PIC17C43/44 PWM Timing	188
PIC17C43/44 RESET, Watchdog Timer, Oscillator	
Start-up Timer and Power-up Timer	186
PIC17C43/44 Timer0 Clock	187
PIC17C43/44 Timer1, Timer2 and Timer3 Clock	187
PIC17C43/44 USART Module Synchronous	
Receive	189
PIC17C43/44 USART Module Synchronous	
Transmission	189
Synchronous Reception	95
Synchronous Transmission	94
Table Read	48
Table Write	46
TMR0	68, 69
TMR0 Read/Write in Timer Mode	70
TMR1, TMR2, and TMR3 in External Clock Mode	80
TMR1, TMR2, and TMR3 in Timer Mode	81
Wake-Up from SLEEP	105
Timing Diagrams and Specifications	155
Timing Parameter Symbology	153
TLRD	44, 139
TLWT	43, 140
TMR0	
16-bit Read	69
16-bit Write	69
Clock Timing	158
Module	68
Operation	68
Overview	65
Prescaler Assignments	69
Read/Write Considerations	69
Read/Write in Timer Mode	70
Timing	68, 69
TMR0 STATUS/Control Register (T0STA)	38
TMR0H	34
TMR0L	34
TMR1	20, 35
8-bit Mode	73
External Clock Input	73
Overview	65
Timer Mode	81
Timing in External Clock Mode	80
Two 8-bit Timer/Counter Mode	73
Using with PWM	75
TMR1CS	71
TMR1IE	23
TMR1IF	24
TMR1ON	72
TMR2	20, 35
8-bit Mode	73
External Clock Input	73
In Timer Mode	81
Timing in External Clock Mode	80
Two 8-bit Timer/Counter Mode	73
Using with PWM	75
TMR2CS	71
TMR2IE	23
TMR2IF	24
TMR2ON	72
TMR3	
Dual Capture1 Register Mode	79
Example, Reading From	80
Example, Writing To	80
External Clock Input	80
In Timer Mode	81
One Capture and One Period Register Mode	78
Overview	65
Reading/Writing	80
Timing in External Clock Mode	80
TMR3CS	71, 77
TMR3H	20, 35
TMR3IE	23
TMR3IF	24, 77
TMR3L	20, 35
TMR3ON	72, 77
$\overline{TO}$	37, 103, 105
Transmit Status and Control Register	83
TRMT	83
TSTFSZ	140
Turning on 16-bit Timer	74
TX9	83
TX9d	83
TXEN	83
TXIE	23
TXIF	24
TXREG	19, 34, 89, 93, 97, 98
TXSTA	19, 34, 92, 96, 98
<b>U</b>	
Upward Compatibility	5
USART	
Asynchronous Master Transmission	90
Asynchronous Mode	89
Asynchronous Receive	91
Asynchronous Transmitter	89
Baud Rate Generator	86
Synchronous Master Mode	93
Synchronous Master Reception	95
Synchronous Master Transmission	93
Synchronous Slave Mode	97
Synchronous Slave Transmit	97
<b>W</b>	
Wake-up from SLEEP	105
Wake-up from SLEEP Through Interrupt	105
Watchdog Timer	99, 103

WDT .....	99, 103
Clearing the WDT .....	103
Normal Timer .....	103
Period .....	103
Programming Considerations .....	103
WDTPS0 .....	99
WDTPS1 .....	99
WREG .....	34

## X

XORLW .....	141
XORWF .....	141

## Z

Z .....	9, 36
Zero (Z) .....	9

## LIST OF EXAMPLES

Example 3-1: Signed Math .....	9
Example 3-2: Instruction Pipeline Flow .....	14
Example 5-1: Saving STATUS and WREG in RAM .....	27
Example 6-1: Indirect Addressing.....	40
Example 7-1: Table Write .....	46
Example 7-2: Table Read.....	48
Example 8-1: 8 x 8 Multiply Routine .....	49
Example 8-2: 8 x 8 Signed Multiply Routine.....	49
Example 8-3: 16 x 16 Multiply Routine .....	50
Example 8-4: 16 x 16 Signed Multiply Routine.....	51
Example 9-1: Initializing PORTB .....	57
Example 9-2: Initializing PORTC .....	58
Example 9-3: Initializing PORTD.....	60
Example 9-4: Initializing PORTE .....	62
Example 9-5: Read Modify Write Instructions on an I/O Port .....	64
Example 11-1: 16-Bit Read .....	69
Example 11-2: 16-Bit Write.....	69
Example 12-1: Sequence to Read Capture Registers.....	78
Example 12-2: Writing to TMR3 .....	80
Example 12-3: Reading from TMR3.....	80
Example 13-1: Calculating Baud Rate Error.....	86
Example F-1: PIC17C42 to Sleep.....	223

## LIST OF FIGURES

Figure 3-1: PIC17C42 Block Diagram .....	10
Figure 3-2: PIC17CR42/42A/43/R43/44 Block Diagram.....	11
Figure 3-3: Clock/Instruction Cycle.....	14
Figure 4-1: Simplified Block Diagram of On-chip Reset Circuit.....	15
Figure 4-2: Time-Out Sequence on Power-Up (MCLR Tied to VDD) .....	17
Figure 4-3: Time-Out Sequence on Power-Up (MCLR <b>NOT</b> Tied to VDD).....	17
Figure 4-4: Slow Rise Time (MCLR Tied to VDD) .....	17
Figure 4-5: Oscillator Start-Up Time .....	18
Figure 4-6: Using On-Chip POR .....	18
Figure 4-7: Brown-out Protection Circuit 1.....	18
Figure 4-8: PIC17C42 External Power-On Reset Circuit (For Slow VDD Power-Up).....	18
Figure 4-9: Brown-out Protection Circuit 2.....	18
Figure 5-1: Interrupt Logic .....	21
Figure 5-2: INTSTA Register (Address: 07h, Unbanked).....	22
Figure 5-3: PIE Register (Address: 17h, Bank 1) .....	23
Figure 5-4: PIR Register (Address: 16h, Bank 1) .....	24
Figure 5-5: INT Pin / T0CKI Pin Interrupt Timing.....	26
Figure 6-1: Program Memory Map and Stack.....	29
Figure 6-2: Memory Map in Different Modes .....	30
Figure 6-3: External Program Memory Access Waveforms .....	31
Figure 6-4: Typical External Program Memory Connection Diagram.....	31
Figure 6-5: PIC17C42 Register File Map.....	33
Figure 6-6: PIC17CR42/42A/43/R43/44 Register File Map.....	33
Figure 6-7: ALUSTA Register (Address: 04h, Unbanked).....	36
Figure 6-8: CPUSTA Register (Address: 06h, Unbanked).....	37
Figure 6-9: TOSTA Register (Address: 05h, Unbanked).....	38
Figure 6-10: Indirect Addressing.....	39
Figure 6-11: Program Counter Operation.....	41

# PIC17C4X

Figure 6-12: Program Counter using The CALL and GOTO Instructions.....	41	Figure 14-3: Crystal Operation, Overtone Crystals (XT OSC Configuration).....	101
Figure 6-13: BSR Operation (PIC17C43/R43/44).....	42	Figure 14-4: External Clock Input Operation (EC OSC Configuration).....	101
Figure 7-1: TLWT Instruction Operation.....	43	Figure 14-5: External Parallel Resonant Crystal Oscillator Circuit.....	102
Figure 7-2: TABLWT Instruction Operation.....	43	Figure 14-6: External Series Resonant Crystal Oscillator Circuit.....	102
Figure 7-3: TLRD Instruction Operation.....	44	Figure 14-7: RC Oscillator Mode.....	102
Figure 7-4: TABLRD Instruction Operation.....	44	Figure 14-8: Watchdog Timer Block Diagram.....	104
Figure 7-5: TABLWT Write Timing (External Memory).....	46	Figure 14-9: Wake-up From Sleep Through Interrupt... ..	105
Figure 7-6: Consecutive TABLWT Write Timing (External Memory).....	47	Figure 15-1: General Format for Instructions.....	108
Figure 7-7: TABLRD Timing.....	48	Figure 15-2: Q Cycle Activity.....	109
Figure 7-8: TABLRD Timing (Consecutive TABLRD Instructions).....	48	Figure 17-1: Parameter Measurement Information.....	154
Figure 9-1: RA0 and RA1 Block Diagram.....	53	Figure 17-2: External Clock Timing.....	155
Figure 9-2: RA2 and RA3 Block Diagram.....	54	Figure 17-3: CLKOUT and I/O Timing.....	156
Figure 9-3: RA4 and RA5 Block Diagram.....	54	Figure 17-4: Reset, Watchdog Timer, Oscillator Start-Up Timer and Power-Up Timer Timing.....	157
Figure 9-4: Block Diagram of RB<7:4> and RB<1:0> Port Pins.....	55	Figure 17-5: Timer0 Clock Timings.....	158
Figure 9-5: Block Diagram of RB3 and RB2 Port Pins..	56	Figure 17-6: Timer1, Timer2, And Timer3 Clock Timings.....	158
Figure 9-6: Block Diagram of RC<7:0> Port Pins.....	58	Figure 17-7: Capture Timings.....	159
Figure 9-7: PORTD Block Diagram (in I/O Port Mode).....	60	Figure 17-8: PWM Timings.....	159
Figure 9-8: PORTE Block Diagram (in I/O Port Mode).....	62	Figure 17-9: USART Module: Synchronous Transmission (Master/Slave) Timing.....	160
Figure 9-9: Successive I/O Operation.....	64	Figure 17-10: USART Module: Synchronous Receive (Master/Slave) Timing.....	160
Figure 11-1: T0STA Register (Address: 05h, Unbanked).....	67	Figure 17-11: Memory Interface Write Timing.....	161
Figure 11-2: Timer0 Module Block Diagram.....	68	Figure 17-12: Memory Interface Read Timing.....	162
Figure 11-3: TMR0 Timing with External Clock (Increment on Falling Edge).....	68	Figure 18-1: Typical RC Oscillator Frequency vs. Temperature.....	163
Figure 11-4: TMR0 Timing: Write High or Low Byte.....	69	Figure 18-2: Typical RC Oscillator Frequency vs. VDD.....	164
Figure 11-5: TMR0 Read/Write in Timer Mode.....	70	Figure 18-3: Typical RC Oscillator Frequency vs. VDD.....	164
Figure 12-1: TCON1 Register (Address: 16h, Bank 3) ...	71	Figure 18-4: Typical RC Oscillator Frequency vs. VDD.....	165
Figure 12-2: TCON2 Register (Address: 17h, Bank 3) ...	72	Figure 18-5: Transconductance (gm) of LF Oscillator vs. VDD.....	166
Figure 12-3: Timer1 and Timer2 in Two 8-bit Timer/Counter Mode.....	73	Figure 18-6: Transconductance (gm) of XT Oscillator vs. VDD.....	166
Figure 12-4: TMR1 and TMR2 in 16-bit Timer/Counter Mode.....	74	Figure 18-7: Typical IDD vs. Frequency (External Clock 25°C).....	167
Figure 12-5: Simplified PWM Block Diagram.....	75	Figure 18-8: Maximum IDD vs. Frequency (External Clock 125°C to -40°C).....	167
Figure 12-6: PWM Output.....	75	Figure 18-9: Typical IPD vs. VDD Watchdog Disabled 25°C.....	168
Figure 12-7: Timer3 with One Capture and One Period Register Block Diagram.....	78	Figure 18-10: Maximum IPD vs. VDD Watchdog Disabled.....	168
Figure 12-8: Timer3 with Two Capture Registers Block Diagram.....	79	Figure 18-11: Typical IPD vs. VDD Watchdog Enabled 25°C.....	169
Figure 12-9: TMR1, TMR2, and TMR3 Operation in External Clock Mode.....	80	Figure 18-12: Maximum IPD vs. VDD Watchdog Enabled.....	169
Figure 12-10: TMR1, TMR2, and TMR3 Operation in Timer Mode.....	81	Figure 18-13: WDT Timer Time-Out Period vs. VDD.....	170
Figure 13-1: TXSTA Register (Address: 15h, Bank 0) ...	83	Figure 18-14: IOH vs. VOH, VDD = 3V.....	170
Figure 13-2: RCSTA Register (Address: 13h, Bank 0) ...	84	Figure 18-15: IOH vs. VOH, VDD = 5V.....	171
Figure 13-3: USART Transmit.....	85	Figure 18-16: IOL vs. VOL, VDD = 3V.....	171
Figure 13-4: USART Receive.....	85	Figure 18-17: IOL vs. VOL, VDD = 5V.....	172
Figure 13-5: Asynchronous Master Transmission.....	90	Figure 18-18: VTH (Input Threshold Voltage) of I/O Pins (TTL) vs. VDD.....	172
Figure 13-6: Asynchronous Master Transmission (Back to Back).....	90	Figure 18-19: VTH, VIL of I/O Pins (Schmitt Trigger) vs. VDD.....	173
Figure 13-7: RX Pin Sampling Scheme.....	91	Figure 18-20: VTH (Input Threshold Voltage) of OSC1 Input (In XT and LF Modes) vs. VDD.....	173
Figure 13-8: Asynchronous Reception.....	92	Figure 19-1: Parameter Measurement Information.....	183
Figure 13-9: Synchronous Transmission.....	94		
Figure 13-10: Synchronous Transmission (Through TXEN).....	94		
Figure 13-11: Synchronous Reception (Master Mode, SREN).....	95		
Figure 14-1: Configuration Word.....	99		
Figure 14-2: Crystal or Ceramic Resonator Operation (XT or LF OSC Configuration).....	100		



Figure 19-2:	External Clock Timing.....	184
Figure 19-3:	CLKOUT and I/O Timing.....	185
Figure 19-4:	Reset, Watchdog Timer, Oscillator Start-Up Timer, and Power-Up Timer Timing.....	186
Figure 19-5:	Timer0 Clock Timings.....	187
Figure 19-6:	Timer1, Timer2, and Timer3 Clock Timings.....	187
Figure 19-7:	Capture Timings.....	188
Figure 19-8:	PWM Timings.....	188
Figure 19-9:	USART Module: Synchronous Transmission (Master/Slave) Timing.....	189
Figure 19-10:	USART Module: Synchronous Receive (Master/Slave) Timing.....	189
Figure 19-11:	Memory Interface Write Timing (Not Supported in PIC17LC4X Devices)...	190
Figure 19-12:	Memory Interface Read Timing (Not Supported in PIC17LC4X Devices)...	191
Figure 20-1:	Typical RC Oscillator Frequency vs. Temperature.....	193
Figure 20-2:	Typical RC Oscillator Frequency vs. VDD.....	194
Figure 20-3:	Typical RC Oscillator Frequency vs. VDD.....	194
Figure 20-4:	Typical RC Oscillator Frequency vs. VDD.....	195
Figure 20-5:	Transconductance (gm) of LF Oscillator vs. VDD.....	196
Figure 20-6:	Transconductance (gm) of XT Oscillator vs. VDD.....	196
Figure 20-7:	Typical IDD vs. Frequency (External Clock 25°C).....	197
Figure 20-8:	Maximum IDD vs. Frequency (External Clock 125°C to -40°C).....	197
Figure 20-9:	Typical IPD vs. VDD Watchdog Disabled 25°C.....	198
Figure 20-10:	Maximum IPD vs. VDD Watchdog Disabled.....	198
Figure 20-11:	Typical IPD vs. VDD Watchdog Enabled 25°C.....	199
Figure 20-12:	Maximum IPD vs. VDD Watchdog Enabled.....	199
Figure 20-13:	WDT Timer Time-Out Period vs. VDD.....	200
Figure 20-14:	IOH vs. VOH, VDD = 3V.....	200
Figure 20-15:	IOH vs. VOH, VDD = 5V.....	201
Figure 20-16:	IOL vs. VOL, VDD = 3V.....	201
Figure 20-17:	IOL vs. VOL, VDD = 5V.....	202
Figure 20-18:	VTH (Input Threshold Voltage) of I/O Pins (TTL) vs. VDD.....	202
Figure 20-19:	VTH, VIL of I/O Pins (Schmitt Trigger) vs. VDD.....	203
Figure 20-20:	VTH (Input Threshold Voltage) of OSC1 Input (In XT and LF Modes) vs. VDD.....	203

## LIST OF TABLES

Table 1-1:	PIC17CXX Family of Devices.....	6
Table 3-1:	Pinout Descriptions.....	12
Table 4-1:	Time-Out in Various Situations.....	16
Table 4-2:	STATUS Bits and Their Significance.....	16
Table 4-3:	Reset Condition for the Program Counter and the CPUSTA Register.....	16
Table 4-4:	Initialization Conditions For Special Function Registers.....	19
Table 5-1:	Interrupt Vectors/Priorities.....	25
Table 6-1:	Mode Memory Access.....	30

Table 6-2:	EPROM Memory Access Time Ordering Suffix.....	31
Table 6-3:	Special Function Registers.....	34
Table 7-1:	Interrupt - Table Write Interaction.....	45
Table 8-1:	Performance Comparison.....	49
Table 9-1:	PORTA Functions.....	54
Table 9-2:	Registers/Bits Associated with PORTA.....	54
Table 9-3:	PORTB Functions.....	57
Table 9-4:	Registers/Bits Associated with PORTB.....	57
Table 9-5:	PORTC Functions.....	59
Table 9-6:	Registers/Bits Associated with PORTC.....	59
Table 9-7:	PORTD Functions.....	61
Table 9-8:	Registers/Bits Associated with PORTD.....	61
Table 9-9:	PORTE Functions.....	63
Table 9-10:	Registers/Bits Associated with PORTE.....	63
Table 11-1:	Registers/Bits Associated with Timer0.....	70
Table 12-1:	Turning On 16-bit Timer.....	74
Table 12-2:	Summary of Timer1 and Timer2 Registers.....	74
Table 12-3:	PWM Frequency vs. Resolution at 25 MHz.....	76
Table 12-4:	Registers/Bits Associated with PWM.....	77
Table 12-5:	Registers Associated with Capture.....	79
Table 12-6:	Summary of TMR1, TMR2, and TMR3 Registers.....	81
Table 13-1:	Baud Rate Formula.....	86
Table 13-2:	Registers Associated with Baud Rate Generator.....	86
Table 13-3:	Baud Rates for Synchronous Mode.....	87
Table 13-4:	Baud Rates for Asynchronous Mode.....	88
Table 13-5:	Registers Associated with Asynchronous Transmission.....	90
Table 13-6:	Registers Associated with Asynchronous Reception.....	92
Table 13-7:	Registers Associated with Synchronous Master Transmission.....	94
Table 13-8:	Registers Associated with Synchronous Master Reception.....	96
Table 13-9:	Registers Associated with Synchronous Slave Transmission.....	98
Table 13-10:	Registers Associated with Synchronous Slave Reception.....	98
Table 14-1:	Configuration Locations.....	100
Table 14-2:	Capacitor Selection for Ceramic Resonators.....	101
Table 14-3:	Capacitor Selection for Crystal Oscillator.....	101
Table 14-4:	Registers/Bits Associated with the Watchdog Timer.....	104
Table 15-1:	Opcode Field Descriptions.....	107
Table 15-2:	PIC17CXX Instruction Set.....	110
Table 16-1:	development tools from microchip.....	146
Table 17-1:	Cross Reference of Device Specs for Oscillator Configurations and Frequencies of Operation (Commercial Devices).....	148
Table 17-2:	External Clock Timing Requirements.....	155
Table 17-3:	CLKOUT and I/O Timing Requirements.....	156
Table 17-4:	Reset, Watchdog Timer, Oscillator Start-Up Timer and Power-Up Timer Requirements.....	157
Table 17-5:	Timer0 Clock Requirements.....	158
Table 17-6:	Timer1, Timer2, and Timer3 Clock Requirements.....	158
Table 17-7:	Capture Requirements.....	159
Table 17-8:	PWM Requirements.....	159

# PIC17C4X

---

Table 17-9:	Serial Port Synchronous Transmission Requirements .....	160
Table 17-10:	Serial Port Synchronous Receive Requirements .....	160
Table 17-11:	Memory Interface Write Requirements .....	161
Table 17-12:	Memory Interface Read Requirements .....	162
Table 18-1:	Pin Capacitance per Package Type .....	163
Table 18-2:	RC Oscillator Frequencies .....	165
Table 19-1:	Cross Reference of Device Specs for Oscillator Configurations and Frequencies of Operation (Commercial Devices) .....	176
Table 19-2:	External Clock Timing Requirements .....	184
Table 19-3:	CLKOUT and I/O Timing Requirements ...	185
Table 19-4:	Reset, Watchdog Timer, Oscillator Start-Up Timer and Power-Up Timer Requirements .....	186
Table 19-5:	Timer0 Clock Requirements .....	187
Table 19-6:	Timer1, Timer2, and Timer3 Clock Requirements .....	187
Table 19-7:	Capture Requirements .....	188
Table 19-8:	PWM Requirements .....	188
Table 19-9:	Synchronous Transmission Requirements .....	189
Table 19-10:	Synchronous Receive Requirements .....	189
Table 19-11:	Memory Interface Write Requirements (Not Supported in PIC17LC4X Devices) ...	190
Table 19-12:	Memory Interface read Requirements (Not Supported in PIC17LC4X Devices) ...	191
Table 20-1:	Pin Capacitance per Package Type .....	193
Table 20-2:	RC Oscillator Frequencies .....	195
Table E-1:	Pin Compatible Devices .....	221

## LIST OF EQUATIONS

Equation 8-1:	16 x 16 Unsigned Multiplication Algorithm .....	50
Equation 8-2:	16 x 16 Signed Multiplication Algorithm .....	51

## ON-LINE SUPPORT

Microchip provides two methods of on-line support. These are the Microchip BBS and the Microchip World Wide Web (WWW) site.

Use Microchip's Bulletin Board Service (BBS) to get current information and help about Microchip products. Microchip provides the BBS communication channel for you to use in extending your technical staff with micro-controller and memory experts.

To provide you with the most responsive service possible, the Microchip systems team monitors the BBS, posts the latest component data and software tool updates, provides technical help and embedded systems insights, and discusses how Microchip products provide project solutions.

The web site, like the BBS, is used by Microchip as a means to make files and information easily available to customers. To view the site, the user must have access to the Internet and a web browser, such as Netscape or Microsoft Explorer. Files are also available for FTP download from our FTP site.

### Connecting to the Microchip Internet Web Site

The Microchip web site is available by using your favorite Internet browser to attach to:

**[www.microchip.com](http://www.microchip.com)**

The file transfer site is available by using an FTP service to connect to:

**[ftp.mchip.com/biz/mchip](ftp://mchip.com/biz/mchip)**

The web site and file transfer site provide a variety of services. Users may download files for the latest Development Tools, Data Sheets, Application Notes, User's Guides, Articles and Sample Programs. A variety of Microchip specific business information is also available, including listings of Microchip sales offices, distributors and factory representatives. Other data available for consideration is:

- Latest Microchip Press Releases
- Technical Support Section with Frequently Asked Questions
- Design Tips
- Device Errata
- Job Postings
- Microchip Consultant Program Member Listing
- Links to other useful web sites related to Microchip Products

### Connecting to the Microchip BBS

Connect worldwide to the Microchip BBS using either the Internet or the CompuServe® communications network.

#### Internet:

You can telnet or ftp to the Microchip BBS at the address:

**[mchipbbs.microchip.com](telnet://mchipbbs.microchip.com)**

### CompuServe Communications Network:

When using the BBS via the CompuServe Network, in most cases, a local call is your only expense. The Microchip BBS connection does not use CompuServe membership services, therefore you do not need CompuServe membership to join Microchip's BBS. There is no charge for connecting to the Microchip BBS.

The procedure to connect will vary slightly from country to country. Please check with your local CompuServe agent for details if you have a problem. CompuServe service allow multiple users various baud rates depending on the local point of access.

The following connect procedure applies in most locations.

1. Set your modem to 8-bit, No parity, and One stop (8N1). This is not the normal CompuServe setting which is 7E1.
2. Dial your local CompuServe access number.
3. Depress the <Enter> key and a garbage string will appear because CompuServe is expecting a 7E1 setting.
4. Type +, depress the <Enter> key and "Host Name:" will appear.
5. Type MCHIPBBS, depress the <Enter> key and you will be connected to the Microchip BBS.

In the United States, to find the CompuServe phone number closest to you, set your modem to 7E1 and dial (800) 848-4480 for 300-2400 baud or (800) 331-7166 for 9600-14400 baud connection. After the system responds with "Host Name:", type NETWORK, depress the <Enter> key and follow CompuServe's directions.

For voice information (or calling from overseas), you may call (614) 723-1550 for your local CompuServe number.

Microchip regularly uses the Microchip BBS to distribute technical information, application notes, source code, errata sheets, bug reports, and interim patches for Microchip systems software products. For each SIG, a moderator monitors, scans, and approves or disapproves files submitted to the SIG. No executable files are accepted from the user community in general to limit the spread of computer viruses.

### Systems Information and Upgrade Hot Line

The Systems Information and Upgrade Line provides system users a listing of the latest versions of all of Microchip's development systems software products. Plus, this line provides information on how customers can receive any currently available upgrade kits. The Hot Line Numbers are:

1-800-755-2345 for U.S. and most of Canada, and  
1-602-786-7302 for the rest of the world.

960513

Trademarks: The Microchip name, logo, PIC, PICSTART, PICMASTER and PRO MATE are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries. FlexROM, MPLAB and fuzzyLAB, are trademarks and SQTP is a service mark of Microchip in the U.S.A.

fuzzyTECH is a registered trademark of Inform Software Corporation. IBM, IBM PC-AT are registered trademarks of International Business Machines Corp. Pentium is a trademark of Intel Corporation. Windows is a trademark and MS-DOS, Microsoft Windows are registered trademarks of Microsoft Corporation. CompuServe is a registered trademark of CompuServe Incorporated.

All other trademarks mentioned herein are the property of their respective companies.

# 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?

---

---



# PIC17C4X

---

NOTES:

# PIC17C4X

---

NOTES:



## WORLDWIDE SALES AND SERVICE

### AMERICAS

#### Corporate Office

Microchip Technology Inc.  
2355 West Chandler Blvd.  
Chandler, AZ 85224-6199  
Tel: 480-786-7200 Fax: 480-786-7277  
Technical Support: 480-786-7627  
Web Address: <http://www.microchip.com>

#### Atlanta

Microchip Technology Inc.  
500 Sugar Mill Road, Suite 200B  
Atlanta, GA 30350  
Tel: 770-640-0034 Fax: 770-640-0307

#### Boston

Microchip Technology Inc.  
5 Mount Royal Avenue  
Marlborough, MA 01752  
Tel: 508-480-9990 Fax: 508-480-8575

#### Chicago

Microchip Technology Inc.  
333 Pierce Road, Suite 180  
Itasca, IL 60143  
Tel: 630-285-0071 Fax: 630-285-0075

#### Dallas

Microchip Technology Inc.  
4570 Westgrove Drive, Suite 160  
Addison, TX 75248  
Tel: 972-818-7423 Fax: 972-818-2924

#### Dayton

Microchip Technology Inc.  
Two Prestige Place, Suite 150  
Miamisburg, OH 45342  
Tel: 937-291-1654 Fax: 937-291-9175

#### Detroit

Microchip Technology Inc.  
Tri-Atria Office Building  
32255 Northwestern Highway, Suite 190  
Farmington Hills, MI 48334  
Tel: 248-538-2250 Fax: 248-538-2260

#### Los Angeles

Microchip Technology Inc.  
18201 Von Karman, Suite 1090  
Irvine, CA 92612  
Tel: 949-263-1888 Fax: 949-263-1338

#### New York

Microchip Technology Inc.  
150 Motor Parkway, Suite 202  
Hauppauge, NY 11788  
Tel: 631-273-5305 Fax: 631-273-5335

#### San Jose

Microchip Technology Inc.  
2107 North First Street, Suite 590  
San Jose, CA 95131  
Tel: 408-436-7950 Fax: 408-436-7955

### AMERICAS (continued)

#### Toronto

Microchip Technology Inc.  
5925 Airport Road, Suite 200  
Mississauga, Ontario L4V 1W1, Canada  
Tel: 905-405-6279 Fax: 905-405-6253

### ASIA/PACIFIC

#### Hong Kong

Microchip Asia Pacific  
Unit 2101, Tower 2  
Metroplaza  
223 Hing Fong Road  
Kwai Fong, N.T., Hong Kong  
Tel: 852-2-401-1200 Fax: 852-2-401-3431

#### Beijing

Microchip Technology, Beijing  
Unit 915, 6 Chaoyangmen Bei Dajie  
Dong Erhuan Road, Dongcheng District  
New China Hong Kong Manhattan Building  
Beijing 100027 PRC  
Tel: 86-10-85282100 Fax: 86-10-85282104

#### India

Microchip Technology Inc.  
India Liaison Office  
No. 6, Legacy, Convent Road  
Bangalore 560 025, India  
Tel: 91-80-229-0061 Fax: 91-80-229-0062

#### Japan

Microchip Technology Intl. Inc.  
Benex S-1 6F  
3-18-20, Shinyokohama  
Kohoku-Ku, Yokohama-shi  
Kanagawa 222-0033 Japan  
Tel: 81-45-471-6166 Fax: 81-45-471-6122

#### Korea

Microchip Technology Korea  
168-1, Youngbo Bldg. 3 Floor  
Samsung-Dong, Kangnam-Ku  
Seoul, Korea  
Tel: 82-2-554-7200 Fax: 82-2-558-5934

#### Shanghai

Microchip Technology  
RM 406 Shanghai Golden Bridge Bldg.  
2077 Yan'an Road West, Hong Qiao District  
Shanghai, PRC 200335  
Tel: 86-21-6275-5700 Fax: 86 21-6275-5060

### ASIA/PACIFIC (continued)

#### Singapore

Microchip Technology Singapore Pte Ltd.  
200 Middle Road  
#07-02 Prime Centre  
Singapore 188980  
Tel: 65-334-8870 Fax: 65-334-8850

#### Taiwan, R.O.C

Microchip Technology Taiwan  
10F-1C 207  
Tung Hua North Road  
Taipei, Taiwan, ROC  
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

### EUROPE

#### United Kingdom

Arizona Microchip Technology Ltd.  
505 Eskdale Road  
Wokingham  
Berkshire, England RG41 5TU  
Tel: 44 118 921 5858 Fax: 44-118 921-5835

#### Denmark

Microchip Technology Denmark ApS  
Regus Business Centre  
Lautrup hof 1-3  
Ballerup DK-2750 Denmark  
Tel: 45 4420 9895 Fax: 45 4420 9910

#### France

Arizona Microchip Technology SARL  
Parc d'Activite du Moulin de Massy  
43 Rue du Saule Trapu  
Batiment A - 1er Etage  
91300 Massy, France  
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

#### Germany

Arizona Microchip Technology GmbH  
Gustav-Heinemann-Ring 125  
D-81739 München, Germany  
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

#### Italy

Arizona Microchip Technology SRL  
Centro Direzionale Colleoni  
Palazzo Taurus 1 V. Le Colleoni 1  
20041 Agrate Brianza  
Milan, Italy  
Tel: 39-039-65791-1 Fax: 39-039-6899883

11/15/99



Microchip received QS-9000 quality system certification for its worldwide headquarters, design and water fabrication facilities in Chandler and Tempe, Arizona in July 1999. The Company's quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOC® code hopping devices, Serial EEPROMs and microperipheral products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001 certified.

All rights reserved. © 1999 Microchip Technology Incorporated. Printed in the USA. 11/99 Printed on recycled paper.

Information contained in this publication regarding device applications and the like is intended for suggestion only and may be superseded by updates. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip's products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights. The Microchip logo and name are registered trademarks of Microchip Technology Inc. in the U.S.A. and other countries. All rights reserved. All other trademarks mentioned herein are the property of their respective companies.