

## 8-Pin, 8-Bit CMOS Microcontroller

### Devices included in this Data Sheet:

PIC12C508 and PIC12C509 are 8-bit microcontrollers packaged in 8-lead packages. They are based on the Enhanced PIC16C5X family.

### High-Performance RISC CPU:

- Only 33 single word instructions to learn
- All instructions are single cycle (1  $\mu$ s) except for program branches which are two-cycle
- Operating speed: DC - 4 MHz clock input  
DC - 1  $\mu$ s instruction cycle

Device	EPROM	RAM
PIC12C508	512 x 12	25
PIC12C509	1024 x 12	41

- 12-bit wide instructions
- 8-bit wide data path
- Seven special function hardware registers
- Two-level deep hardware stack
- Direct, indirect and relative addressing modes for data and instructions
- Internal 4 MHz RC oscillator with programmable calibration
- In-circuit serial programming

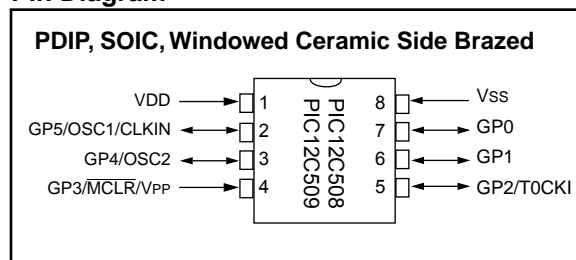
### Peripheral Features:

- 8-bit real time clock/counter (TMR0) with 8-bit programmable prescaler
- Power-On Reset (POR)
- Device Reset Timer (DRT)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code-protection
- Power saving SLEEP mode
- Wake-up from SLEEP on pin change
- Internal weak pull-ups on I/O pins
- Internal pull-up on  $\overline{\text{MCLR}}$  pin
- Selectable oscillator options:
  - INTRC: Internal 4 MHz RC oscillator
  - EXTRC: External low-cost RC oscillator
  - XT: Standard crystal/resonator
  - LP: Power saving, low frequency crystal

### CMOS Technology:

- Low power, high speed CMOS EPROM technology
- Fully static design
- Wide operating voltage range:
  - Commercial: 2.5V to 5.5V
  - Industrial: 2.5V to 5.5V
  - Extended: 3.0V to 5.5V
- Low power consumption
  - < 2 mA @ 5V, 4 MHz
  - 15  $\mu$ A typical @ 3V, 32 KHz
  - < 1  $\mu$ A typical standby current

### Pin Diagram



# PIC12C5XX

---

---

## TABLE OF CONTENTS

1.0	General Description .....	3
2.0	PIC12C5XX Device Varieties .....	5
3.0	Architectural Overview .....	7
4.0	Memory Organization .....	11
5.0	I/O Port .....	19
6.0	Timer0 Module and TMR0 Register .....	21
7.0	Special Features of the CPU .....	25
8.0	Instruction Set Summary .....	37
9.0	Development Support .....	49
10.0	Electrical Characteristics - PIC12C5XX .....	53
11.0	DC and AC Characteristics - PIC12C5XX .....	61
12.0	Packaging Information .....	65
	PIC12C5XX Product Identification System .....	73

### *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, 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.

## 1.0 GENERAL DESCRIPTION

The PIC12C5XX from Microchip Technology is a family of low-cost, high performance, 8-bit, fully static, EPROM/ROM-based CMOS microcontrollers. It employs a RISC architecture with only 33 single word/single cycle instructions. All instructions are single cycle (1  $\mu$ s) except for program branches which take two cycles. The PIC12C5XX delivers performance an order of magnitude higher than its competitors in the same price category. The 12-bit wide instructions are highly symmetrical resulting in 2:1 code compression over other 8-bit microcontrollers in its class. The easy to use and easy to remember instruction set reduces development time significantly.

The PIC12C5XX products are equipped with special features that reduce system cost and power requirements. The Power-On Reset (POR) and Device Reset Timer (DRT) eliminate the need for external reset circuitry. There are four oscillator configurations to choose from, including INTRC internal oscillator mode and the power-saving LP (Low Power) oscillator. Power saving SLEEP mode, Watchdog Timer and code protection features improve system cost, power and reliability.

The PIC12C5XX are available in the cost-effective One-Time-Programmable (OTP) versions which are suitable for production in any volume. The customer can take full advantage of Microchip's price leadership in OTP microcontrollers while benefiting from the OTP's flexibility.

The PIC12C5XX products are supported by a full-featured macro assembler, a software simulator, an in-circuit emulator, a 'C' compiler, fuzzy logic support tools, a low-cost development programmer, and a full featured programmer. All the tools are supported on IBM<sup>®</sup> PC and compatible machines.

## 1.1 Applications

The PIC12C5XX series fits perfectly in applications ranging from personal care appliances and security systems to low-power remote transmitters/receivers. The EPROM technology makes customizing application programs (transmitter codes, appliance settings, receiver frequencies, etc.) extremely fast and convenient. The small footprint packages, for through hole or surface mounting, make this microcontroller series perfect for applications with space limitations. Low-cost, low-power, high performance, ease of use and I/O flexibility make the PIC12C5XX series very versatile even in areas where no microcontroller use has been considered before (e.g., timer functions, replacement of "glue" logic and PLD's in larger systems, coprocessor applications).

# PIC12C5XX

---

---

**TABLE 1-1: PIC12C5XX FAMILY OF DEVICES**

		<b>PIC12C508</b>	<b>PIC12C509</b>
<b>Clock</b>	Maximum Frequency of Operation (MHz)	4	4
<b>Memory</b>	EPROM Program Memory (x12 words)	512	1024
	Data Memory (bytes)	25	41
<b>Peripherals</b>	Timer Module(s)	TMR0	TMR0
<b>Features</b>	Wake-up from SLEEP on pin change	Yes	Yes
	I/O Pins	5	5
	Input Pins	1	1
	Internal Pull-ups	Yes	Yes
	Voltage Range (Volts)	2.5-5.5	2.5-5.5
	In-Circuit Serial Programming	Yes	Yes
	Number of Instructions	33	33
	Packages	8-pin DIP, SOIC	8-pin DIP, SOIC

All PIC12C5XX devices have Power-on Reset, selectable Watchdog Timer, selectable code protect and high I/O current capability. All PIC12C5XX devices use serial programming with data pin GP1 and clock pin GP0.

## 2.0 PIC12C5XX DEVICE VARIETIES

A variety of packaging options are available. Depending on application and production requirements, the proper device option can be selected using the information in this section. When placing orders, please use the PIC12C5XX Product Identification System at the back of this data sheet to specify the correct part number.

### 2.1 UV Erasable Devices

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

The UV erasable version can be erased and reprogrammed to any of the configuration modes.

<p><b>Note:</b> Please note that erasing the device will also erase the pre-programmed internal calibration value for the internal oscillator. The calibration value must be saved prior to erasing the part.</p>
---

Microchip's PICSTART<sup>®</sup> PLUS and PRO MATE<sup>™</sup> programmers all support programming of the PIC12C5XX. Third party programmers also are available; refer to the *Microchip 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 who need the flexibility for frequent code updates or small volume applications.

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 fuse options already programmed by the factory. Certain code and prototype verification procedures do 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.

# PIC12C5XX

---

---

NOTES:

## 3.0 ARCHITECTURAL OVERVIEW

The high performance of the PIC12C5XX family can be attributed to a number of architectural features commonly found in RISC microprocessors. To begin with, the PIC12C5XX uses a Harvard architecture in which program and data are accessed on separate buses. This improves bandwidth over traditional von Neumann architecture where program and data are fetched on the same bus. Separating program and data memory further allows instructions to be sized differently than the 8-bit wide data word. Instruction opcodes are 12-bits wide making it possible to have all single word instructions. A 12-bit wide program memory access bus fetches a 12-bit instruction in a single cycle. A two-stage pipeline overlaps fetch and execution of instructions. Consequently, all instructions (33) execute in a single cycle (1 $\mu$ s @ 4MHz) except for program branches.

The PIC12C508 address 512 x 12 of program memory, the PIC12C509 addresses 1K x 12 of program memory. All program memory is internal.

The PIC12C5XX can directly or indirectly address its register files and data memory. All special function registers including the program counter are mapped in the data memory. The PIC12C5XX has a highly 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 PIC12C5XX simple yet efficient. In addition, the learning curve is reduced significantly.

The PIC12C5XX device contains 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. In two-operand instructions, typically one operand is the W (working) register. The other operand is either a file register or an immediate constant. In single operand instructions, the operand is either the W register or a file register.

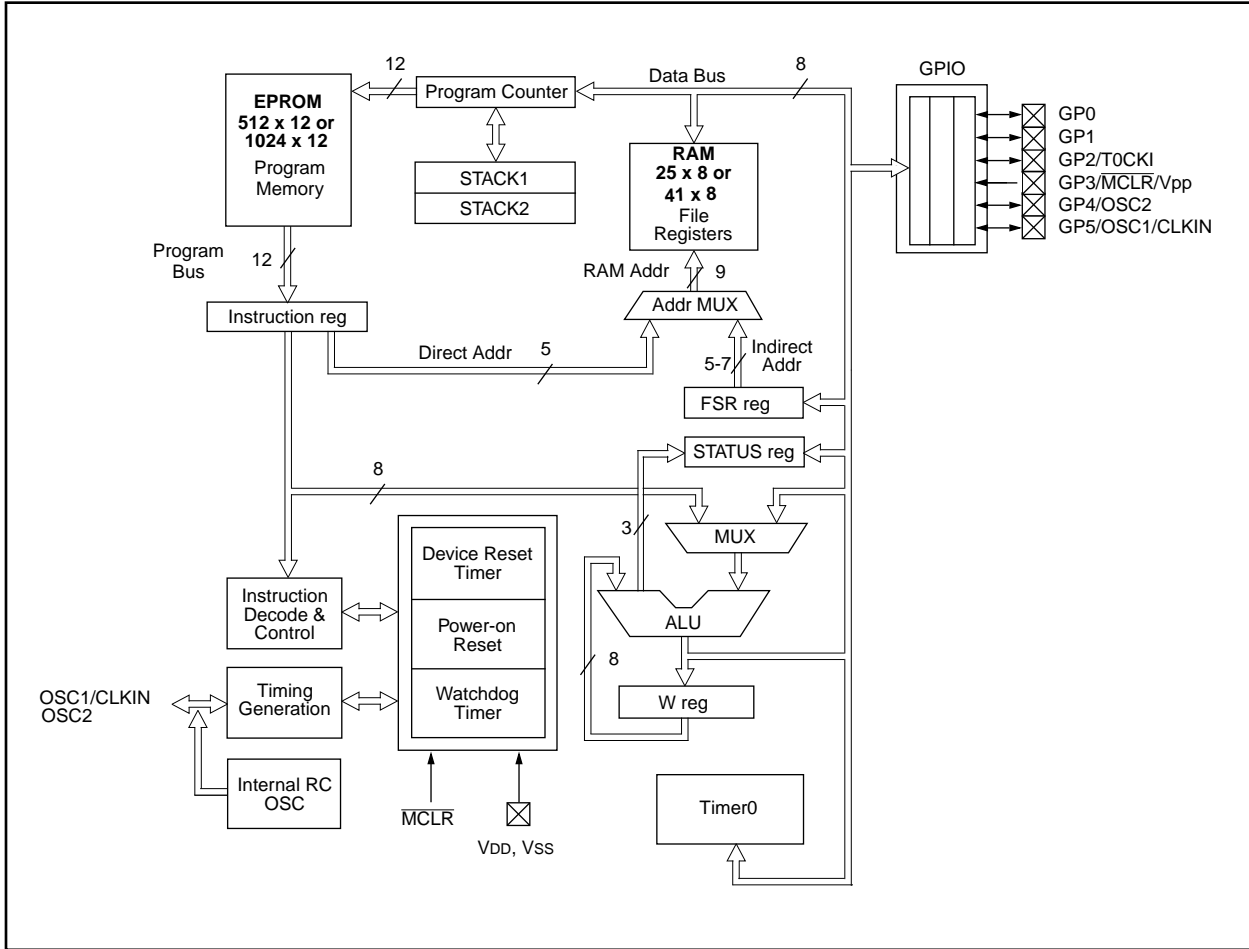
The W register is an 8-bit working register used for ALU operations. It is not an addressable register.

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 SUBWF and ADDWF instructions for examples.

A simplified block diagram is shown in Figure 3-1, with the corresponding device pins described in Table 3-1.

# PIC12C5XX

FIGURE 3-1: PIC12C5XX BLOCK DIAGRAM





**TABLE 3-1: PIC12C5XX PINOUT DESCRIPTION**

Name	DIP Pin #	SOIC Pin #	I/O/P Type	Buffer Type	Description
GP0	7	7	I/O	TTL/ST	Bi-directional I/O port/ serial programming data. Can be software programmed for internal weak pull-up and wake-up from SLEEP on pin change. This buffer is a Schmitt Trigger input when used in serial programming mode.
GP1	6	6	I/O	TTL/ST	Bi-directional I/O port/ serial programming clock. Can be software programmed for internal weak pull-up and wake-up from SLEEP on pin change. This buffer is a Schmitt Trigger input when used in serial programming mode.
GP2/T0CKI	5	5	I/O	ST	Bi-directional I/O port. Can be configured as T0CKI.
GP3/MCLR/VPP	4	4	I	TTL	Input port/master clear (reset) input/programming voltage input. When configured as MCLR, this pin is an active low reset to the device. Voltage on MCLR/VPP must not exceed VDD during normal device operation. Can be software programmed for internal weak pull-up and wake-up from SLEEP on pin change. Weak pull-up always on if configured as MCLR
GP4/OSC2	3	3	I/O	TTL	Bi-directional I/O port/oscillator crystal output. Connections to crystal or resonator in crystal oscillator mode (XT and LP modes only, GPIO in other modes).
GP5/OSC1/CLKIN	2	2	I/O	TTL/ST	Bidirectional IO port/oscillator crystal input/external clock source input (GPIO in Internal RC mode only, OSC1 in all other oscillator modes). TTL input when GPIO, ST input in external RC oscillator mode.
VDD	1	1	P	—	Positive supply for logic and I/O pins
VSS	8	8	P	—	Ground reference for logic and I/O pins

Legend: I = input, O = output, I/O = input/output, P = power, — = not used, TTL = TTL input, ST = Schmitt Trigger input

# PIC12C5XX

## 3.1 Clocking Scheme/Instruction Cycle

The clock input (OSC1/CLKIN pin) is internally divided by four to generate four non-overlapping quadrature clocks namely Q1, Q2, Q3 and Q4. Internally, the program counter is incremented every Q1, and the instruction is fetched from program memory and latched into instruction register in Q4. It is decoded and executed during the following Q1 through Q4. The clocks and instruction execution flow is shown in Figure 3-2 and Example 3-1.

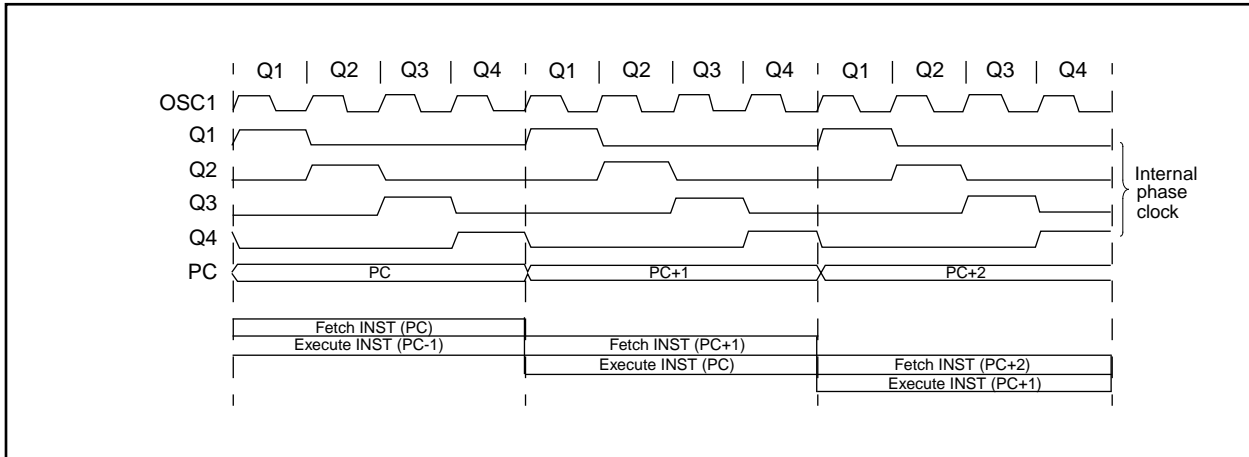
## 3.2 Instruction Flow/Pipelining

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 3-1).

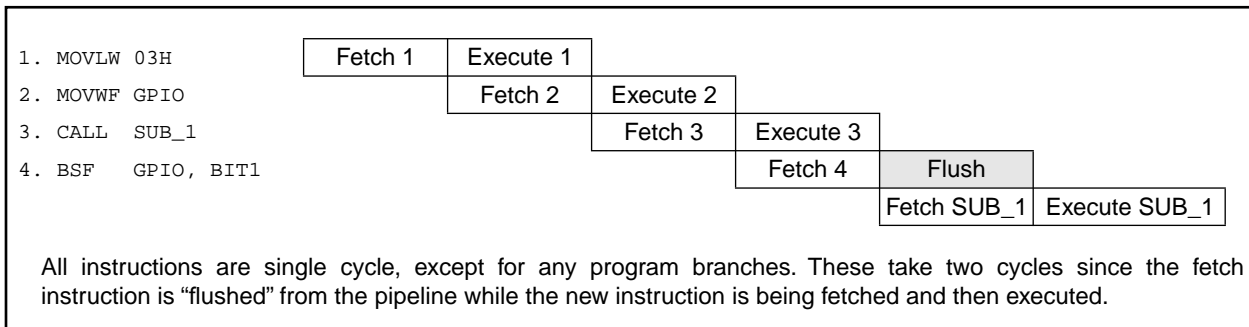
A fetch cycle begins with the program counter (PC) incrementing in Q1.

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 3-2: CLOCK/INSTRUCTION CYCLE**



**EXAMPLE 3-1: INSTRUCTION PIPELINE FLOW**



## 4.0 MEMORY ORGANIZATION

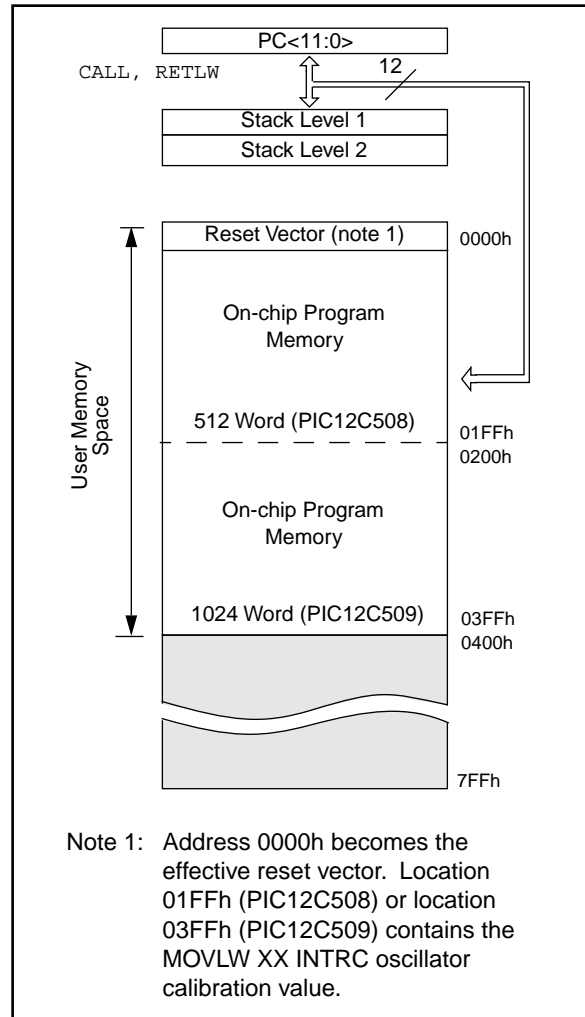
PIC12C5XX memory is organized into program memory and data memory. For devices with more than 512 bytes of program memory, a paging scheme is used. Program memory pages are accessed using one STATUS register bit. For the PIC12C509 with a data memory register file of more than 32 registers, a banking scheme is used. Data memory banks are accessed using the File Select Register (FSR).

### 4.1 Program Memory Organization

The PIC12C508 and PIC12C509 each have a 12-bit Program Counter (PC) capable of addressing a 2K x 12 program memory space.

Only the first 512 x 12 (0000h-01FFh) for the PIC12C508 and 1K x 12 (0000h-03FFh) for the PIC12C509 are physically implemented. Refer to Figure 4-1. Accessing a location above these boundaries will cause a wrap-around within the first 512 x 12 space (PIC12C508) or 1K x 12 space (PIC12C509). The effective reset vector is at 000h, (see Figure 4-1). Location 01FFh (PIC12C508) or location 03FFh (PIC12C509) contains the internal clock oscillator calibration value. This value should never be overwritten.

**FIGURE 4-1: PROGRAM MEMORY MAP AND STACK FOR THE PIC12C5XX**



# PIC12C5XX

## 4.2 Data Memory Organization

Data memory is composed of registers, or bytes of RAM. Therefore, data memory for a device is specified by its register file. The register file is divided into two functional groups: special function registers and general purpose registers.

The special function registers include the TMR0 register, the Program Counter (PC), the Status Register, the I/O registers (ports), and the File Select Register (FSR). In addition, special purpose registers are used to control the I/O port configuration and prescaler options.

The general purpose registers are used for data and control information under command of the instructions.

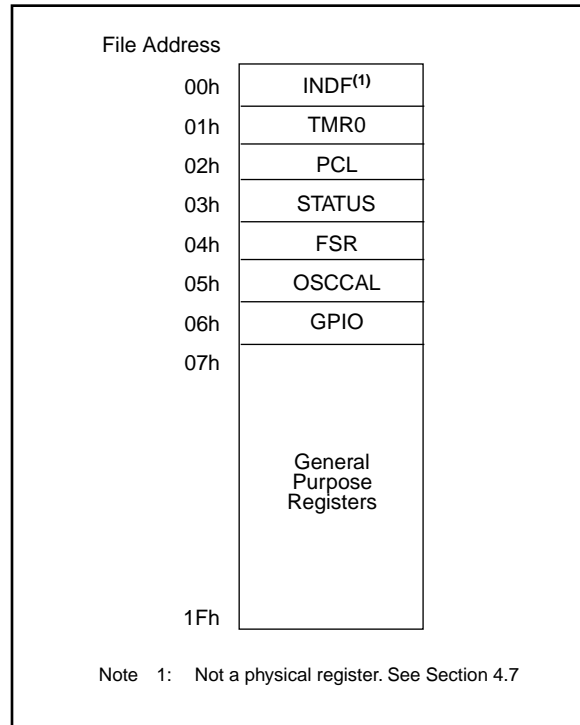
For the PIC12C508, the register file is composed of 7 special function registers and 25 general purpose registers (Figure 4-2).

For the PIC12C509, the register file is composed of 7 special function registers, 25 general purpose registers, and 16 general purpose registers that may be addressed using a banking scheme (Figure 4-3).

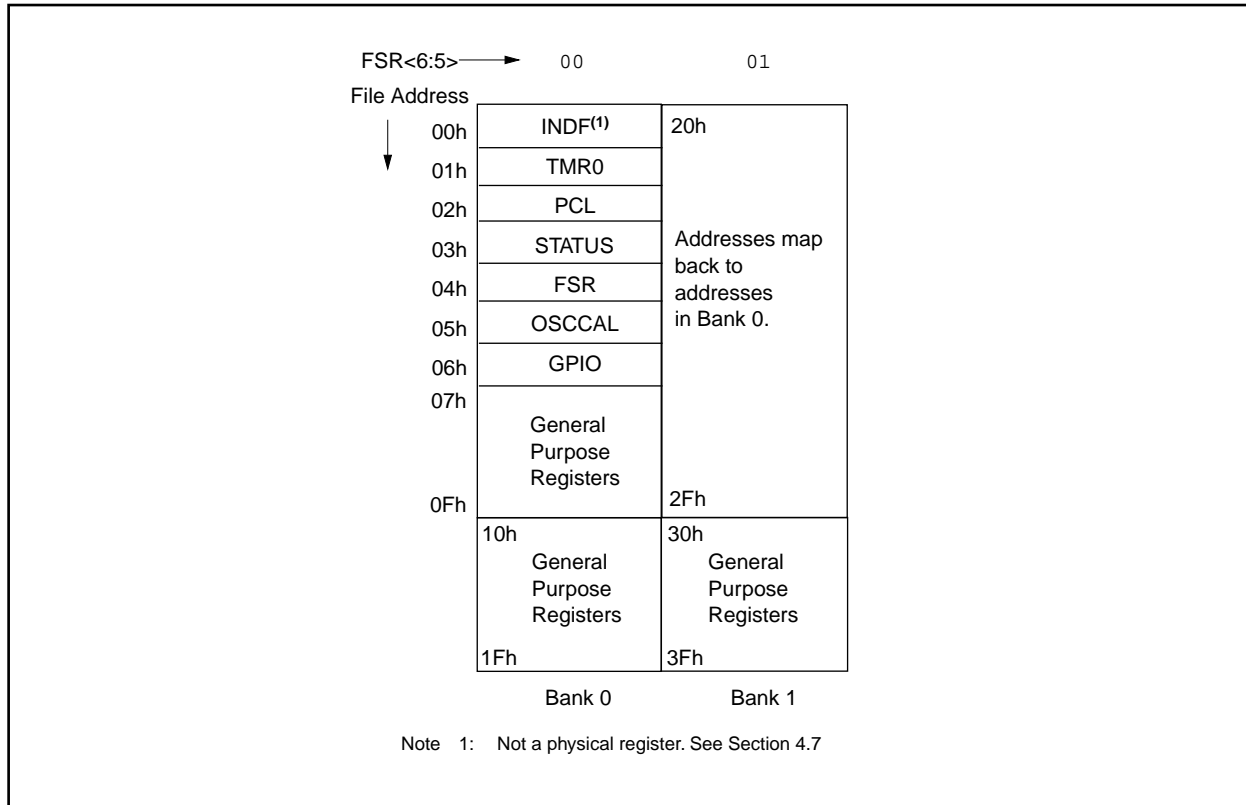
### 4.2.1 GENERAL PURPOSE REGISTER FILE

The general purpose register file is accessed either directly or indirectly through the file select register FSR (Section 4.7).

**FIGURE 4-2: PIC12C508 REGISTER FILE MAP**



**FIGURE 4-3: PIC12C509 REGISTER FILE MAP**



## 4.2.2 SPECIAL FUNCTION REGISTERS

The Special Function Registers (SFRs) are registers used by the CPU and peripheral functions to control the operation of the device (Table 4-1).

The special registers can be classified into two sets. The special function registers associated with the “core” functions are described in this section. Those related to the operation of the peripheral features are described in the section for each peripheral feature.

**TABLE 4-1: SPECIAL FUNCTION REGISTER (SFR) SUMMARY**

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 MCLR and WDT Reset	Value on Wake-up on Pin Change
N/A	TRIS	I/O control registers								--11 1111	--11 1111	--11 1111
N/A	OPTION	Contains control bits to configure Timer0, Timer0/WDT prescaler, wake-up on change, and weak pull-ups								1111 1111	1111 1111	1111 1111
00h	INDF	Uses contents of FSR to address data memory (not a physical register)								xxxx xxxx	uuuu uuuu	uuuu uuuu
01h	TMR0	8-bit real-time clock/counter								xxxx xxxx	uuuu uuuu	uuuu uuuu
02h <sup>(1)</sup>	PCL	Low order 8 bits of PC								1111 1111	1111 1111	1111 1111
03h	STATUS	GPWUF	—	PA0	$\overline{TO}$	PD	Z	DC	C	0001 1xxx	000q quuu	100q quuu
04h	FSR (12C508)	Indirect data memory address pointer								111x xxxx	111u uuuu	111u uuuu
04h	FSR (12C509)	Indirect data memory address pointer								110x xxxx	11uu uuuu	11uu uuuu
05h	OSCCAL	CAL7	CAL6	CAL5	CAL4	—	—	—	—	0111 ----	uuuu ----	uuuu ----
06h	GPIO	—	—	GP5	GP4	GP3	GP2	GP1	GP0	--xx xxxx	--uu uuuu	--uu uuuu

Legend: Shaded boxes = unimplemented or unused, — = unimplemented, read as '0' (if applicable)  
 x = unknown, u = unchanged, q = see the tables in Section 7.7 for possible values.

Note 1: The upper byte of the Program Counter is not directly accessible. See Section 4.5 for an explanation of how to access these bits.



## 4.4 OPTION Register

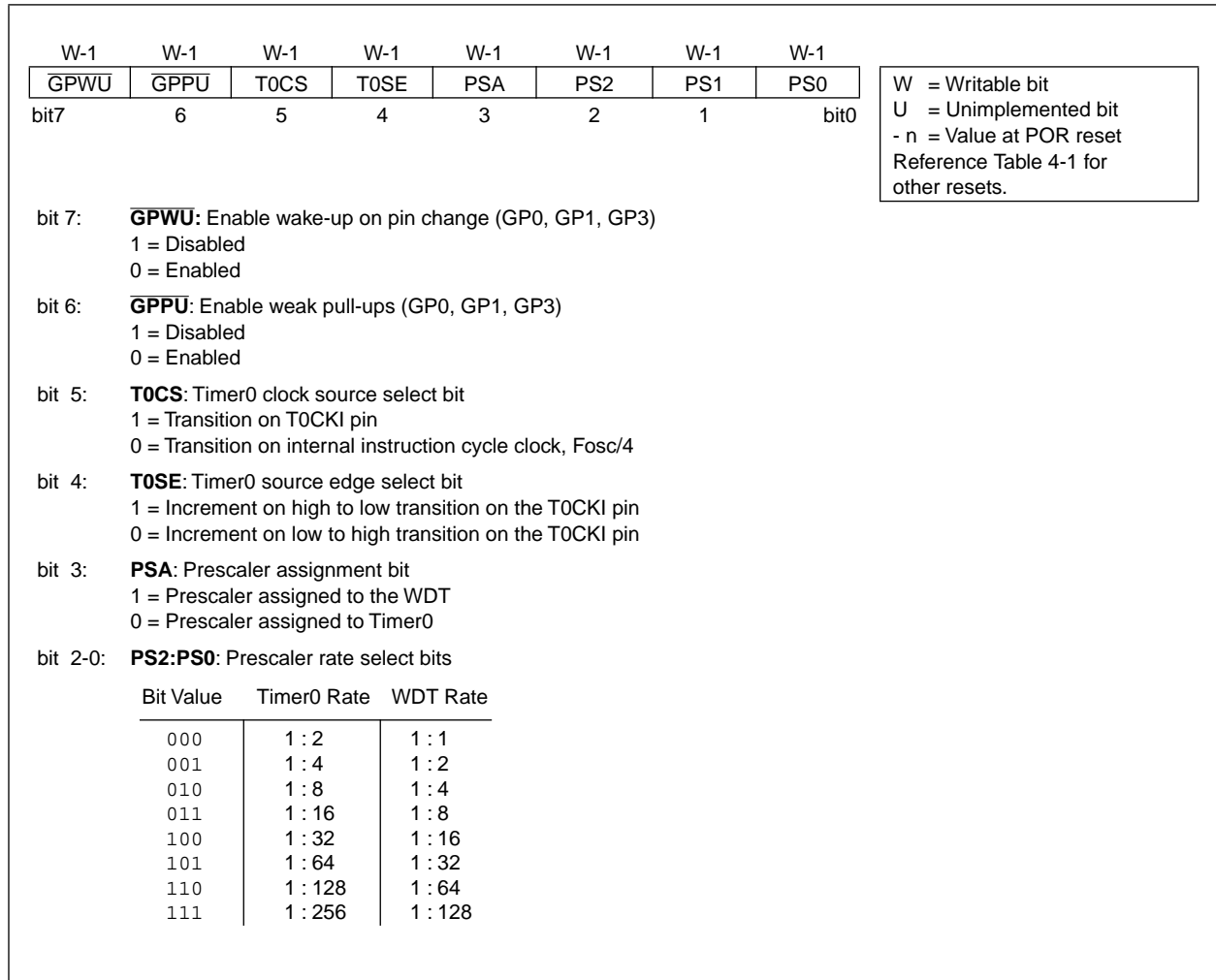
The OPTION register is a 8-bit wide, write-only register which contains various control bits to configure the Timer0/WDT prescaler and Timer0.

By executing the OPTION instruction, the contents of the W register will be transferred to the OPTION register. A RESET sets the OPTION<7:0> bits.

**Note:** If TRIS bit is set to '0', the wake-up on change and pull-up functions are disabled for that pin; i.e., note that TRIS overrides OPTION control of  $\overline{\text{GPPU}}$  and  $\overline{\text{GPWU}}$ .

**Note:** If the T0CS bit is set to '1', GP2 is forced to be an input even if TRIS GP2 = '0'.

**FIGURE 4-5: OPTION REGISTER**



# PIC12C5XX

## 4.5 Program Counter

As a program instruction is executed, the Program Counter (PC) will contain the address of the next program instruction to be executed. The PC value is increased by one every instruction cycle, unless an instruction changes the PC.

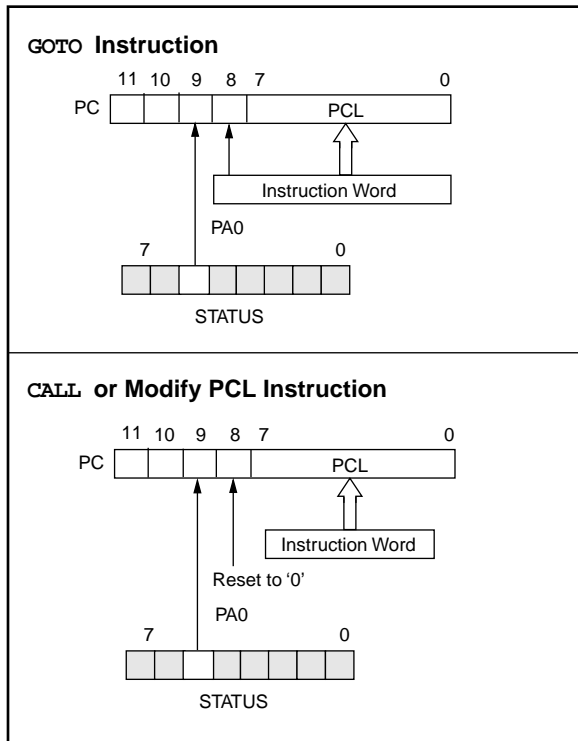
For a `GOTO` instruction, bits 8:0 of the PC are provided by the `GOTO` instruction word. The PC Latch (PCL) is mapped to `PC<7:0>`. Bit 5 of the STATUS register provides page information to bit 9 of the PC (Figure 4-6).

For a `CALL` instruction, or any instruction where the PCL is the destination, bits 7:0 of the PC again are provided by the instruction word. However, `PC<8>` does not come from the instruction word, but is always cleared (Figure 4-6).

Instructions where the PCL is the destination, or Modify PCL instructions, include `MOVWF PC`, `ADDWF PC`, and `BSF PC, 5`.

**Note:** Because `PC<8>` is cleared in the `CALL` instruction, or any Modify PCL instruction, all subroutine calls or computed jumps are limited to the first 256 locations of any program memory page (512 words long).

**FIGURE 4-6: LOADING OF PC BRANCH INSTRUCTIONS - PIC12C508/C509**



### 4.5.1 EFFECTS OF RESET

The Program Counter is set upon a RESET, which means that the PC addresses the last location in the last page i.e., the oscillator calibration instruction. After executing `MOVLW XX`, the PC will roll over to location 00h, and begin executing user code.

The STATUS register page preselect bits are cleared upon a RESET, which means that page 0 is pre-selected.

Therefore, upon a RESET, a `GOTO` instruction will automatically cause the program to jump to page 0 until the value of the page bits is altered.

## 4.6 Stack

PIC12C5XX devices have a 12-bit wide hardware push/pop stack.

A `CALL` instruction will *push* the current value of stack 1 into stack 2 and then push the current program counter value, incremented by one, into stack level 1. If more than two sequential `CALL`'s are executed, only the most recent two return addresses are stored.

A `RETLW` instruction will *pop* the contents of stack level 1 into the program counter and then copy stack level 2 contents into level 1. If more than two sequential `RETLW`'s are executed, the stack will be filled with the address previously stored in level 2. Note that the W register will be loaded with the literal value specified in the instruction. This is particularly useful for the implementation of data look-up tables within the program memory.



## 4.7 Indirect Data Addressing; INDF and FSR Registers

The INDF register is not a physical register. Addressing INDF actually addresses the register whose address is contained in the FSR register (FSR is a *pointer*). This is indirect addressing.

### EXAMPLE 4-1: INDIRECT ADDRESSING

- Register file 07 contains the value 10h
- Register file 08 contains the value 0Ah
- Load the value 07 into the FSR register
- A read of the INDF register will return the value of 10h
- Increment the value of the FSR register by one (FSR = 08)
- A read of the INDF register now will return the value of 0Ah.

Reading INDF itself indirectly (FSR = 0) will produce 00h. Writing to the INDF register indirectly results in a no-operation (although STATUS bits may be affected).

A simple program to clear RAM locations 10h-1Fh using indirect addressing is shown in Example 4-2.

### EXAMPLE 4-2: HOW TO CLEAR RAM USING INDIRECT ADDRESSING

```

movlw 0x10 ;initialize pointer
movwf FSR ; to RAM
NEXT   clrfs INDF ;clear INDF register
       incf FSR,F ;inc pointer
       btfsc FSR,4 ;all done?
       goto NEXT ;NO, clear next

CONTINUE
       : ;YES, continue
    
```

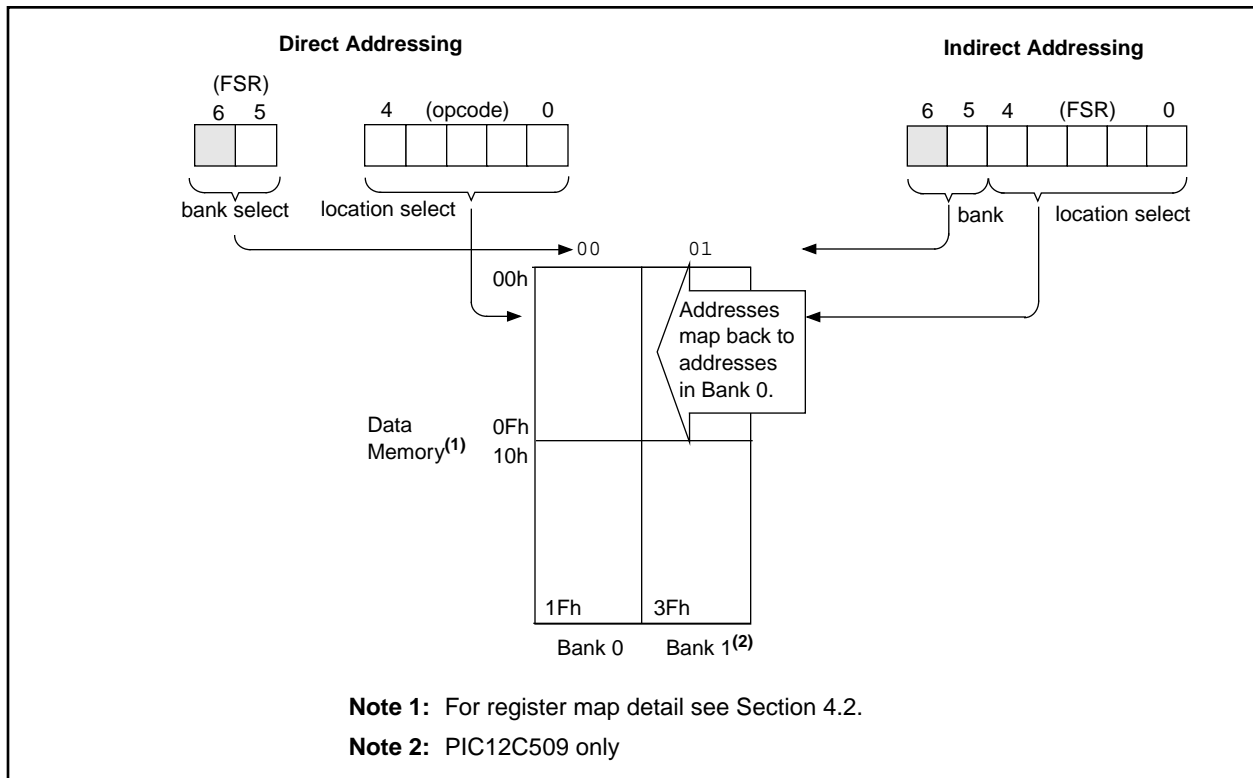
The FSR is a 5-bit wide register. It is used in conjunction with the INDF register to indirectly address the data memory area.

The FSR<4:0> bits are used to select data memory addresses 00h to 1Fh.

**PIC12C508:** Does not use banking. FSR<6:5> are unimplemented and read as '1's.

**PIC12C509:** Uses FSR<5>. Selects between bank 0 and bank 1. FSR<6> is unimplemented, read as '1'.

FIGURE 4-7: DIRECT/INDIRECT ADDRESSING



# PIC12C5XX

---

NOTES:

## 5.0 I/O PORT

As with any other register, the I/O register can be written and read under program control. However, read instructions (e.g., `MOVF GPIO, W`) always read the I/O pins independent of the pin's input/output modes. On RESET, all I/O ports are defined as input (inputs are at hi-impedance) since the I/O control registers are all set.

### 5.1 GPIO

GPIO is an 8-bit I/O register. Only the low order 6 bits are used (GP5:GP0). Bits 7 and 6 are unimplemented and read as '0's. Please note that GP3 is an input only pin. The configuration word can set several I/O's to alternate functions. When acting as alternate functions the pins will read as '0' during port read. Pins GP0, GP1, and GP3 can be configured with weak pull-ups and also with wake-up on change. The wake-up on change and weak pull-up functions are not pin selectable. If pin 4 is configured as  $\overline{\text{MCLR}}$ , weak pull-up is always on and wake-up on change for this pin is not enabled.

### 5.2 TRIS Register

The output driver control register is loaded with the contents of the W register by executing the `TRIS f` instruction. A '1' from a TRIS register bit puts the corresponding output driver in a hi-impedance mode. A '0' puts the contents of the output data latch on the selected pins, enabling the output buffer. The exceptions are GP3 which is input only and GP2 which may be controlled by the option register, see Section 4.4.

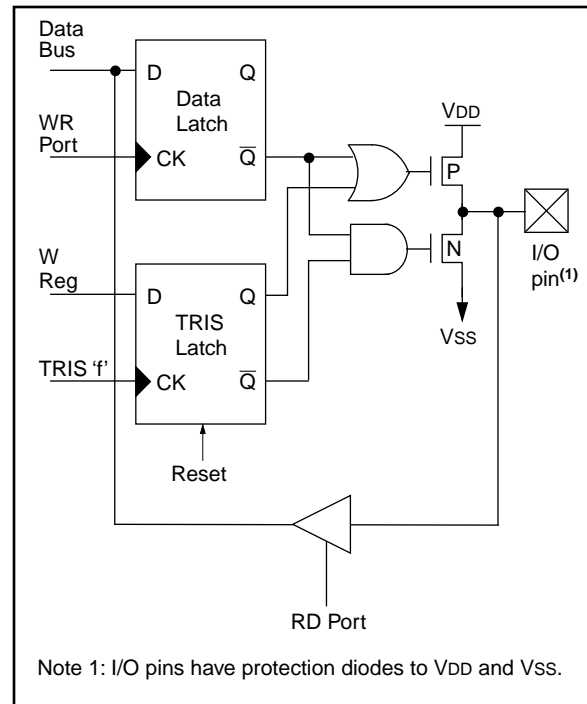
**Note:** A read of the ports reads the pins, not the output data latches. That is, if an output driver on a pin is enabled and driven high, but the external system is holding it low, a read of the port will indicate that the pin is low.

The TRIS registers are "write-only" and are set (output drivers disabled) upon RESET.

## 5.3 I/O Interfacing

The equivalent circuit for an I/O port pin is shown in Figure 5-1. All port pins, except GP3 which is input only, may be used for both input and output operations. For input operations these ports are non-latching. Any input must be present until read by an input instruction (e.g., `MOVF GPIO, W`). The outputs are latched and remain unchanged until the output latch is rewritten. To use a port pin as output, the corresponding direction control bit in TRIS must be cleared (= 0). For use as an input, the corresponding TRIS bit must be set. Any I/O pin (except GP3) can be programmed individually as input or output.

**FIGURE 5-1: EQUIVALENT CIRCUIT FOR A SINGLE I/O PIN**



**TABLE 5-1: SUMMARY OF PORT 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 MCLR and WDT Reset	Value on Wake-up on Pin Change
N/A	TRIS	I/O control registers								--11 1111	--11 1111	--11 1111
N/A	OPTION	GPWU	GPPU	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111	1111 1111
03H	STATUS	GPWUF	—	PA0	$\overline{\text{TO}}$	$\overline{\text{PD}}$	Z	DC	C	0001 1xxx	000q quuu	100q quuu
06h	GPIO	—	—	GP5	GP4	GP3	GP2	GP1	GP0	--xx xxxx	--uu uuuu	--uu uuuu

Legend: Shaded cells not used by Port Registers, read as '0', — = unimplemented, read as '0', x = unknown, u = unchanged, q = see tables in Section 7.7 for possible values.

## 5.4 I/O Programming Considerations

### 5.4.1 BI-DIRECTIONAL I/O PORTS

Some instructions operate internally as read followed by write operations. The BCF and BSF instructions, for example, read the entire port into the CPU, execute the bit operation and re-write the result. Caution must be used when these instructions are applied to a port where one or more pins are used as input/outputs. For example, a BSF operation on bit5 of GPIO will cause all eight bits of GPIO to be read into the CPU, bit5 to be set and the GPIO value to be written to the output latches. If another bit of GPIO is used as a bi-directional I/O pin (say 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 rewritten 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.

Example 5-1 shows the effect of two sequential read-modify-write instructions (e.g., BCF, BSF, etc.) on an I/O port.

A pin actively outputting a high or a low should not be driven from external devices at the same time in order to change the level on this pin ("wired-or", "wired-and"). The resulting high output currents may damage the chip.

### EXAMPLE 5-1: READ-MODIFY-WRITE INSTRUCTIONS ON AN I/O PORT

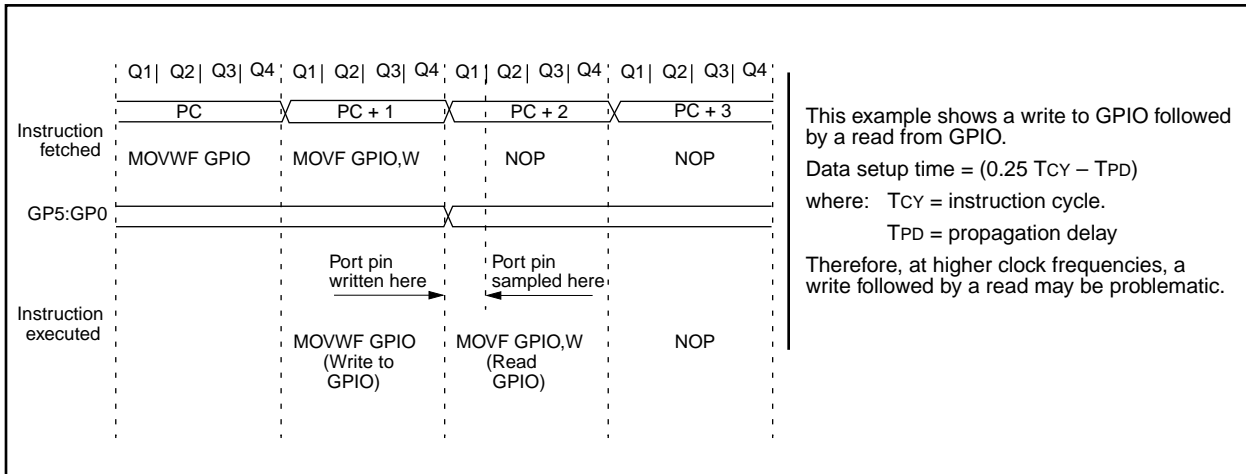
```

;Initial GPIO Settings
; GPIO<5:3> Inputs
; GPIO<2:0> Outputs
;
;
;           GPIO latch  GPIO pins
;           -----  -----
BCF  GPIO, 5  ;--01 -ppp  --11 pppp
BCF  GPIO, 4  ;--10 -ppp  --11 pppp
MOVLW 007h   ;
TRIS GPIO    ;--10 -ppp  --11 pppp
;
;Note that the user may have expected the pin
;values to be --00 pppp. The 2nd BCF caused
;GP5 to be latched as the pin value (High).
    
```

### 5.4.2 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 5-2). 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 allow the pin voltage to stabilize (load dependent) before the next instruction, which causes that file to be read into the CPU, is executed. 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 5-2: SUCCESSIVE I/O OPERATION



## 6.0 TIMER0 MODULE AND TMR0 REGISTER

The Timer0 module has the following features:

- 8-bit timer/counter register, TMR0
  - Readable and writable
- 8-bit software programmable prescaler
- Internal or external clock select
  - Edge select for external clock

Figure 6-1 is a simplified block diagram of the Timer0 module.

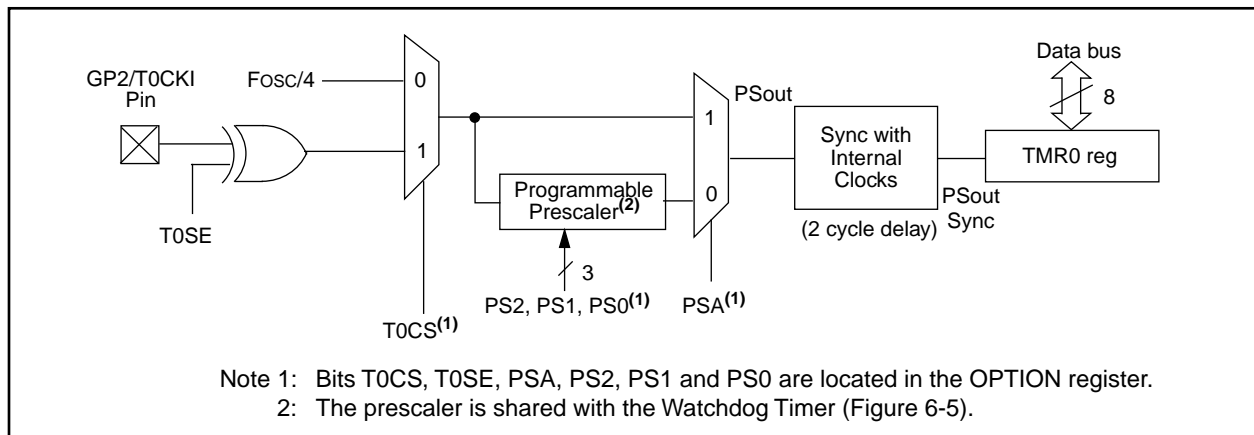
Timer mode is selected by clearing the T0CS bit (OPTION<5>). In timer mode, the Timer0 module will increment every instruction cycle (without prescaler). If TMR0 register is written, the increment is inhibited for the following two cycles (Figure 6-2 and Figure 6-3). The user can work around this by writing an adjusted value to the TMR0 register.

Counter mode is selected by setting the T0CS bit (OPTION<5>). In this mode, Timer0 will increment either on every rising or falling edge of pin T0CKI. The T0SE bit (OPTION<4>) determines the source edge. Clearing the T0SE bit selects the rising edge. Restrictions on the external clock input are discussed in detail in Section 6.1.

The prescaler may be used by either the Timer0 module or the Watchdog Timer, but not both. The prescaler assignment is controlled in software by the control bit PSA (OPTION<3>). Clearing the PSA bit will assign the prescaler to Timer0. The prescaler is not readable or writable. When the prescaler is assigned to the Timer0 module, prescale values of 1:2, 1:4, ..., 1:256 are selectable. Section 6.2 details the operation of the prescaler.

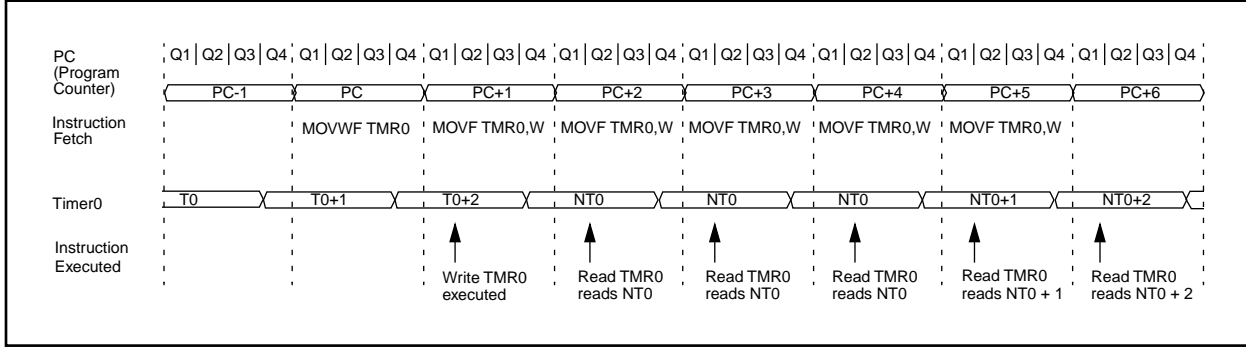
A summary of registers associated with the Timer0 module is found in Table 6-1.

**FIGURE 6-1: TIMER0 BLOCK DIAGRAM**

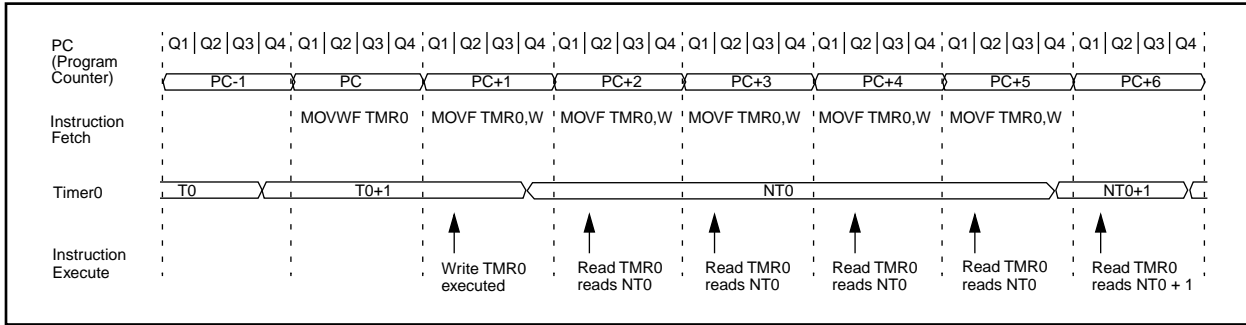


# PIC12C5XX

**FIGURE 6-2: TIMER0 TIMING: INTERNAL CLOCK/NO PRESCALE**



**FIGURE 6-3: TIMER0 TIMING: INTERNAL CLOCK/PRESCALE 1:2**



**TABLE 6-1: REGISTERS 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 MCLR and WDT Reset	Value on Wake-up on Pin Change
01h	TMR0	Timer0 - 8-bit real-time clock/counter								xxxx xxxx	uuuu uuuu	uuuu uuuu
N/A	OPTION	GPWU	GPPU	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111	1111 1111
N/A	TRIS	I/O control registers								--11 1111	--11 1111	--11 1111

Legend: Shaded cells not used by Timer0, - = unimplemented, x = unknown, u = unchanged,

## 6.1 Using Timer0 with an External Clock

When an external clock input is used for Timer0, it must meet certain requirements. The external clock requirement is due to internal phase clock (TOSC) synchronization. Also, there is a delay in the actual incrementing of Timer0 after synchronization.

### 6.1.1 EXTERNAL CLOCK SYNCHRONIZATION

When no prescaler is used, the external clock input is the same as the prescaler output. The synchronization of T0CKI with the internal phase clocks is accomplished by sampling the prescaler output on the Q2 and Q4 cycles of the internal phase clocks (Figure 6-4). Therefore, it is necessary for T0CKI to be high for at least 2TOSC (and a small RC delay of 20 ns) and low for at least 2TOSC (and a small RC delay of 20 ns). Refer to the electrical specification of the desired device.

When a prescaler is used, the external clock input is divided by the asynchronous ripple counter-type prescaler so that the prescaler output is symmetrical. For the external clock to meet the sampling requirement, the ripple counter must be taken into account. Therefore, it is necessary for T0CKI to have a period of at least 4TOSC (and a small RC delay of 40 ns) divided by the prescaler value. The only requirement on T0CKI high and low time is that they do not violate the minimum pulse width requirement of 10 ns. Refer to parameters 40, 41 and 42 in the electrical specification of the desired device.

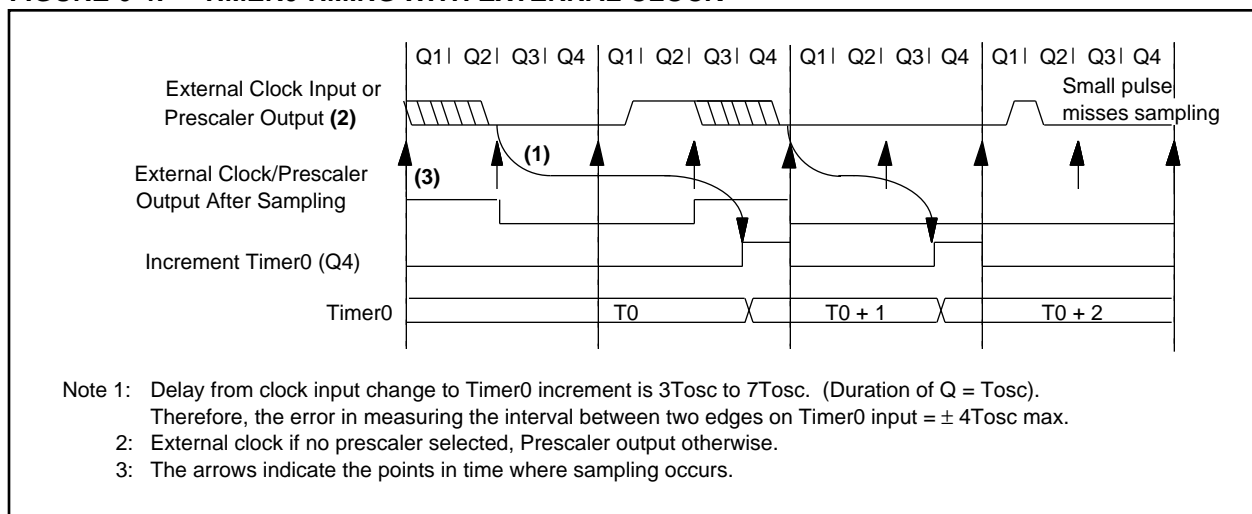
### 6.1.2 TIMER0 INCREMENT DELAY

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 the Timer0 module is actually incremented. Figure 6-4 shows the delay from the external clock edge to the timer incrementing.

### 6.1.3 OPTION REGISTER EFFECT ON GP2 TRIS

If the option register is set to read TIMER0 from the pin, the port is forced to an input regardless of the TRIS register setting.

**FIGURE 6-4: TIMER0 TIMING WITH EXTERNAL CLOCK**



# PIC12C5XX

## 6.2 Prescaler

An 8-bit counter is available as a prescaler for the Timer0 module, or as a postscaler for the Watchdog Timer (WDT), respectively (Section 7.6). For simplicity, this counter is being referred to as “prescaler” throughout this data sheet. Note that the prescaler may be used by either the Timer0 module or the WDT, but not both. Thus, a prescaler assignment for the Timer0 module means that there is no prescaler for the WDT, and vice-versa.

The PSA and PS2:PS0 bits (OPTION<3:0>) determine prescaler assignment and prescale ratio.

When assigned to the Timer0 module, all instructions writing to the TMR0 register (e.g., CLRF 1, MOVWF 1, BSF 1,x, etc.) will clear the prescaler. When assigned to WDT, a CLRWDT instruction will clear the prescaler along with the WDT. The prescaler is neither readable nor writable. On a RESET, the prescaler contains all '0's.

### 6.2.1 SWITCHING PRESCALER ASSIGNMENT

The prescaler assignment is fully under software control (i.e., it can be changed “on the fly” during program execution). To avoid an unintended device RESET, the following instruction sequence (Example 6-1) must be executed when changing the prescaler assignment from Timer0 to the WDT.

### EXAMPLE 6-1: CHANGING PRESCALER (TIMER0→WDT)

```

1.CLRWDT           ;Clear WDT
2.CLRF   TMR0     ;Clear TMR0 & Prescaler
3.MOVLW  '00xx1111'b ;These 3 lines (5, 6, 7)
4.OPTION           ; are required only if
                   ; desired
5.CLRWDT           ;PS<2:0> are 000 or 001
6.MOVLW  '00xx1xxx'b ;Set Postscaler to
7.OPTION           ; desired WDT rate
    
```

To change prescaler from the WDT to the Timer0 module, use the sequence shown in Example 6-2. This sequence must be used even if the WDT is disabled. A CLRWDT instruction should be executed before switching the prescaler.

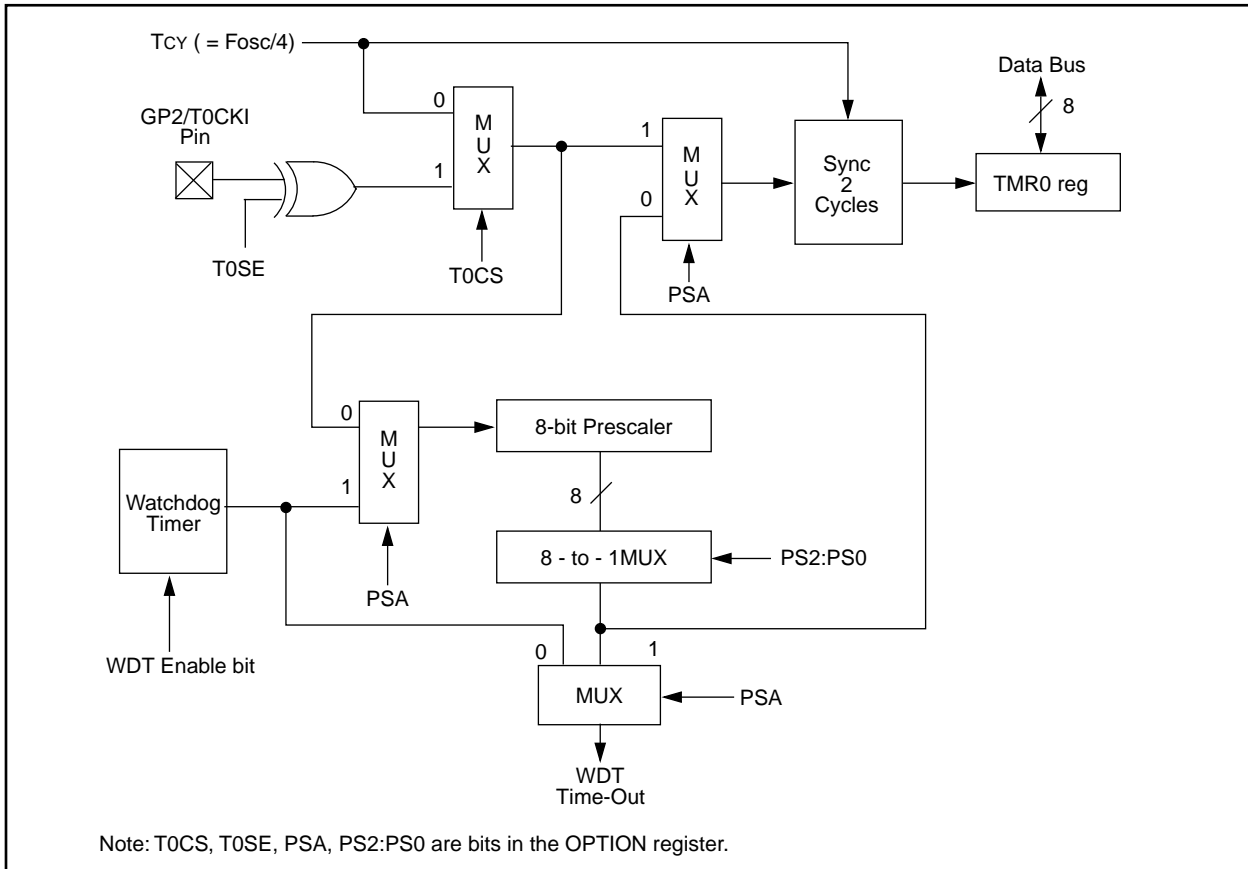
### EXAMPLE 6-2: CHANGING PRESCALER (WDT→TIMER0)

```

CLRWDT           ;Clear WDT and
                ;prescaler
MOVLW  'xxxx0xxx' ;Select TMR0, new
                ;prescale value and
                ;clock source

OPTION
    
```

FIGURE 6-5: BLOCK DIAGRAM OF THE TIMER0/WDT PRESCALER





## 7.0 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 PIC12C5XX family of microcontrollers 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 features are:

- Oscillator selection
- Reset
  - Power-On Reset (POR)
  - Device Reset Timer (DRT)
  - Wake-up from SLEEP on pin change
- Watchdog Timer (WDT)
- SLEEP
- Code protection
- ID locations
- In-circuit Serial Programming

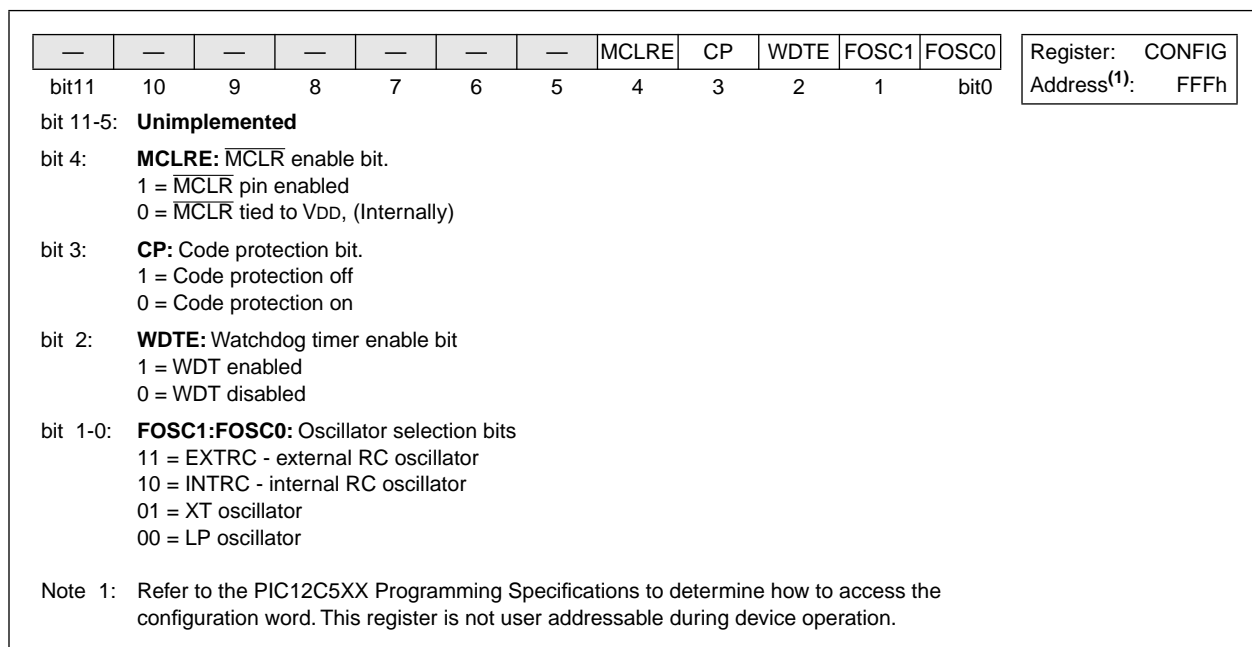
The PIC12C5XX has a Watchdog Timer which can be shut off only through configuration bit WDTE. It runs off of its own RC oscillator for added reliability. If using XT or LP selectable oscillator options, there is always an 18 ms (nominal) delay provided by the Device Reset Timer (DRT), intended to keep the chip in reset until the crystal oscillator is stable. If using INTRC or EXTRC there is an 18 ms delay only on VDD power-up. With this timer 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-up from SLEEP through a change on input pins or through a Watchdog Timer time-out. Several oscillator options are also made available to allow the part to fit the application, including an internal 4 MHz oscillator. The EXTRC oscillator option saves system cost while the LP crystal option saves power. A set of configuration bits are used to select various options.

### 7.1 Configuration Bits

The PIC12C5XX configuration word consists of 5 bits. Configuration bits can be programmed to select various device configurations. Two bits are for the selection of the oscillator type, one bit is the Watchdog Timer enable bit, and one bit is the  $\overline{\text{MCLR}}$  enable bit. One bit is the code protection bit (Figure 7-1).

**FIGURE 7-1: CONFIGURATION WORD FOR PIC12C508 OR PIC12C509**



# PIC12C5XX

## 7.2 Oscillator Configurations

### 7.2.1 OSCILLATOR TYPES

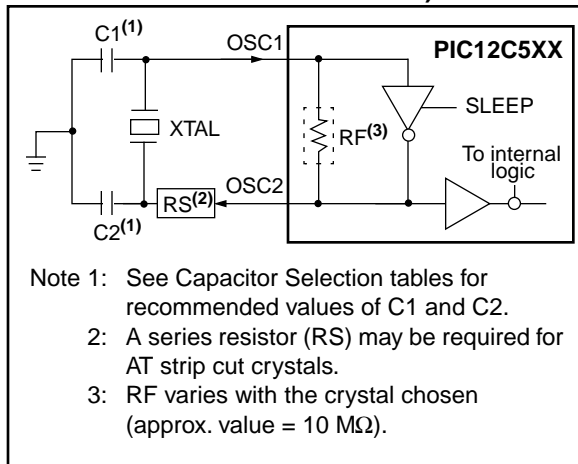
The PIC12C5XX can be operated in four different oscillator modes. The user can program two configuration bits (FOSC1:FOSC0) to select one of these four modes:

- LP: Low Power Crystal
- XT: Crystal/Resonator
- INTRC: Internal 4 MHz Oscillator
- EXTRC: External Resistor/Capacitor

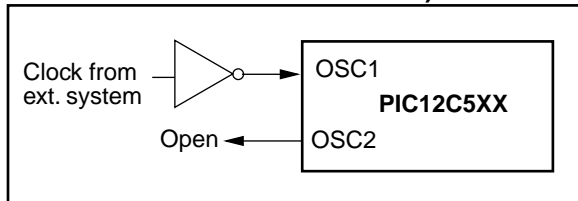
### 7.2.2 CRYSTAL OSCILLATOR / CERAMIC RESONATORS

In XT or LP modes, a crystal or ceramic resonator is connected to the GP5/OSC1/CLKIN and GP4/OSC2 pins to establish oscillation (Figure 7-2). The PIC12C5XX 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. When in XT or LP modes, the device can have an external clock source drive the GP5/OSC1/CLKIN pin (Figure 7-3).

**FIGURE 7-2: CRYSTAL OPERATION (OR CERAMIC RESONATOR) (XT OR LP OSC CONFIGURATION)**



**FIGURE 7-3: EXTERNAL CLOCK INPUT OPERATION (XT OR LP OSC CONFIGURATION)**



**TABLE 7-1: CAPACITOR SELECTION FOR CERAMIC RESONATORS - PIC12C5XX**

Osc Type	Resonator Freq	Cap. Range C1	Cap. Range C2
XT	4.0 MHz	30 pF	30 pF

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.

**TABLE 7-2: CAPACITOR SELECTION FOR CRYSTAL OSCILLATOR - PIC12C5XX**

Osc Type	Resonator Freq	Cap. Range C1	Cap. Range C2
LP	32 kHz <sup>(1)</sup>	15 pF	15 pF
XT	200 kHz	47-68 pF	47-68 pF
	1 MHz	15 pF	15 pF
	4 MHz	15 pF	15 pF

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

These values are for design guidance only. Rs may be required in XT mode to avoid overdriving crystals with low drive level specification. Since each crystal has its own characteristics, the user should consult the crystal manufacturer for appropriate values of external components.

## 7.2.3 EXTERNAL CRYSTAL OSCILLATOR CIRCUIT

Either a prepackaged oscillator or a simple oscillator circuit with TTL gates can be used as an external crystal oscillator circuit. 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 parallel resonance, or one with series resonance.

Figure 7-4 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 potentiometers bias the 74AS04 in the linear region. This circuit could be used for external oscillator designs.

**FIGURE 7-4: EXTERNAL PARALLEL RESONANT CRYSTAL OSCILLATOR CIRCUIT**

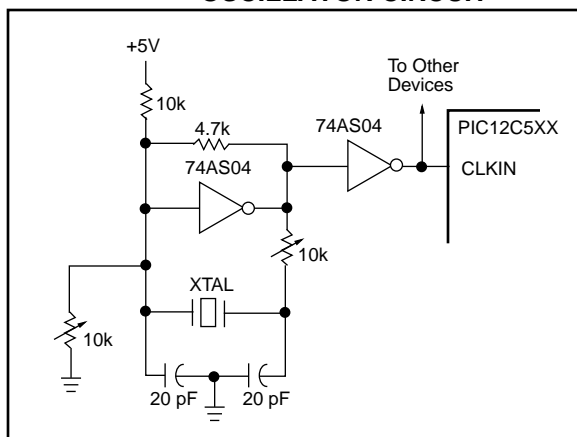
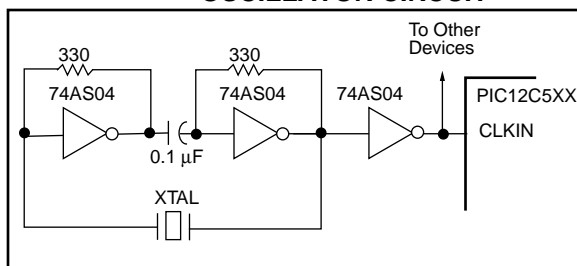


Figure 7-5 shows a series resonant oscillator circuit. This circuit is also designed to use the fundamental frequency of the crystal. The inverter performs a 180-degree phase shift in a series resonant oscillator circuit. The 330 Ω resistors provide the negative feedback to bias the inverters in their linear region.

**FIGURE 7-5: EXTERNAL SERIES RESONANT CRYSTAL OSCILLATOR CIRCUIT**



## 7.2.4 EXTERNAL RC OSCILLATOR

For timing insensitive applications, the RC device option offers additional cost savings. The 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, the 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 the 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.

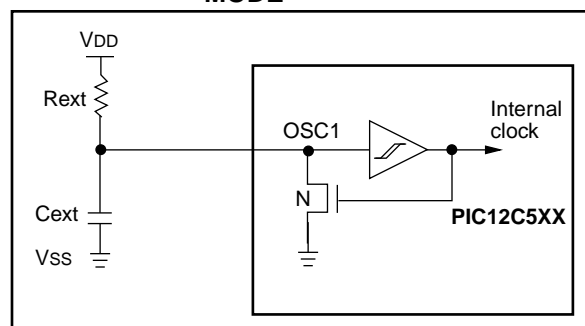
Figure 7-6 shows how the R/C combination is connected to the PIC12C5XX. For  $R_{ext}$  values below 2.2 kΩ, the oscillator operation may become unstable, or stop completely. For very high  $R_{ext}$  values (e.g., 1 MΩ) the oscillator becomes sensitive to noise, humidity and leakage. Thus, we recommend keeping  $R_{ext}$  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 no or small external capacitance, the oscillation frequency can vary dramatically due to changes in external capacitances, such as PCB trace capacitance or package lead frame capacitance.

The Electrical Specifications sections show 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).

Also, see the Electrical Specifications sections 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.

**FIGURE 7-6: EXTERNAL RC OSCILLATOR MODE**



# PIC12C5XX

## 7.2.5 INTERNAL 4 MHz RC OSCILLATOR

The internal RC oscillator provides a fixed 4 MHz (nominal) system clock.

In addition, a calibration instruction is programmed into the top of memory which contains the calibration value for the internal RC oscillator. This value is programmed as a `MOVLW XX` instruction where `XX` is the calibration value, and is placed at the reset vector. This will load the `W` register with the calibration value upon reset and the PC will then roll over to the users program at address `0x000`. The user then has the option of writing the value to the `OSCCAL` Register (`05h`) or ignoring it.

`OSCCAL`, when written to with the calibration value, will “trim” the internal oscillator to remove process variation from the oscillator frequency. The upper 4 bits of the register are used to allow for future, longer bit length calibration schemes. Writing a larger value in this location yields a higher clock speed.

**Note:** Please note that erasing the device will also erase the pre-programmed internal calibration value for the internal oscillator. The calibration value must be saved prior to erasing the part.

## 7.3 RESET

The device differentiates between various kinds of reset:

- a) Power on reset (POR)
- b)  $\overline{\text{MCLR}}$  reset during normal operation
- c)  $\overline{\text{MCLR}}$  reset during SLEEP
- d) WDT time-out reset during normal operation
- e) WDT time-out reset during SLEEP
- f) Wake-up from SLEEP on pin change

Some registers are not reset in any way; they are unknown on POR and unchanged in any other reset. Most other registers are reset to “reset state” on power-on reset (POR), on  $\overline{\text{MCLR}}$ , WDT or wake-up on pin change reset during normal operation. They are not affected by a WDT reset during SLEEP or  $\overline{\text{MCLR}}$  reset during SLEEP, since these resets are viewed as resumption of normal operation. The exceptions to this are  $\overline{\text{TO}}$ ,  $\overline{\text{PD}}$ , and `GPWUF` bits. They are set or cleared differently in different reset situations. These bits are used in software to determine the nature of reset. See Table 7-3 for a full description of reset states of all registers.

**TABLE 7-3: RESET CONDITIONS FOR REGISTERS**

Register	Address	Power-on Reset	MCLR Reset WDT time-out Wake-up on Pin Change
W	—	q q q q x x x x (1)	q q q q u u u u (1)
INDF	00h	x x x x x x x x	u u u u u u u u
TMR0	01h	x x x x x x x x	u u u u u u u u
PC	02h	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1
STATUS	03h	0 0 0 1 1 x x x	? 0 0 ? ? u u u (2)
FSR (12C508)	04h	1 1 1 x x x x x	1 1 1 u u u u u
FSR (12C509)	04h	1 1 0 x x x x x	1 1 u u u u u u
OSCCAL	05h	0 1 1 1 - - - -	u u u u - - - -
GPIO	06h	- - x x x x x x	- - u u u u u u
OPTION	—	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1
TRIS	—	- - 1 1 1 1 1 1	- - 1 1 1 1 1 1

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

**Note 1:** Bits <7:4> of `W` register contain oscillator calibration values due to `MOVLW XX` instruction at top of memory.

**Note 2:** See Table 7-7 for reset value for specific conditions

**TABLE 7-4: RESET CONDITION FOR SPECIAL REGISTERS**

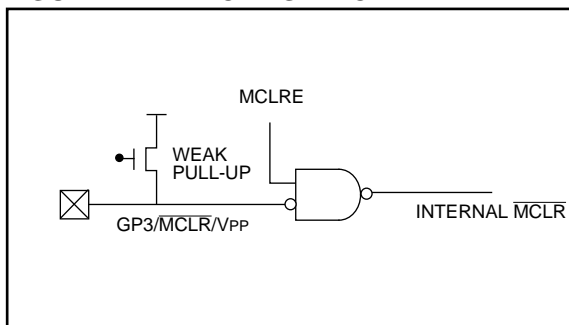
	STATUS Addr: 03h	PCL Addr: 02h
Power on reset	0001 1xxx	1111 1111
MCLR reset during normal operation	000u uuuu	1111 1111
MCLR reset during SLEEP	0001 0uuuu	1111 1111
WDT reset during SLEEP	0000 0uuu	1111 1111
WDT reset normal operation	0000 1uuu	1111 1111
Wake-up from SLEEP on pin change	1001 0uuu	1111 1111

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

### 7.3.1 MCLR ENABLE

This configuration bit when unprogrammed (left in the '1' state) enables the external MCLR function. When programmed, the MCLR function is tied to the internal VDD, and the pin is assigned to be a GPIO. See Figure 7-7.

**FIGURE 7-7: MCLR SELECT**



### 7.4 Power-On Reset (POR)

The PIC12C5XX family incorporates on-chip Power-On Reset (POR) circuitry which provides an internal chip reset for most power-up situations.

A Power-on Reset pulse is generated on-chip when VDD rise is detected (in the range of 1.5V - 2.1V). To take advantage of the internal POR, program the GP3/MCLR/VPP pin as MCLR and tie directly to VDD or program the pin as GP3. An internal weak pull-up resistor is implemented using a transistor. Refer to Table 10-6 for the pull-up resistor ranges. This will eliminate external RC components usually needed to create a Power-on Reset. A maximum rise time for VDD is specified. See Electrical Specifications for details.

When the device starts normal operation (exits the reset condition), device operating parameters (voltage, frequency, temperature, ...) must be met to ensure operation. If these conditions are not met, the device must be held in reset until the operating parameters are met.

A simplified block diagram of the on-chip Power-On Reset circuit is shown in Figure 7-8.

The Power-On Reset circuit and the Device Reset Timer (Section 7.5) circuit are closely related. On power-up, the reset latch is set and the DRT is reset. The DRT timer begins counting once it detects MCLR to be high. After the time-out period, which is typically 18 ms, it will reset the reset latch and thus end the on-chip reset signal.

A power-up example where MCLR is held low is shown in Figure 7-9. VDD is allowed to rise and stabilize before bringing MCLR high. The chip will actually come out of reset TDRT msec after MCLR goes high.

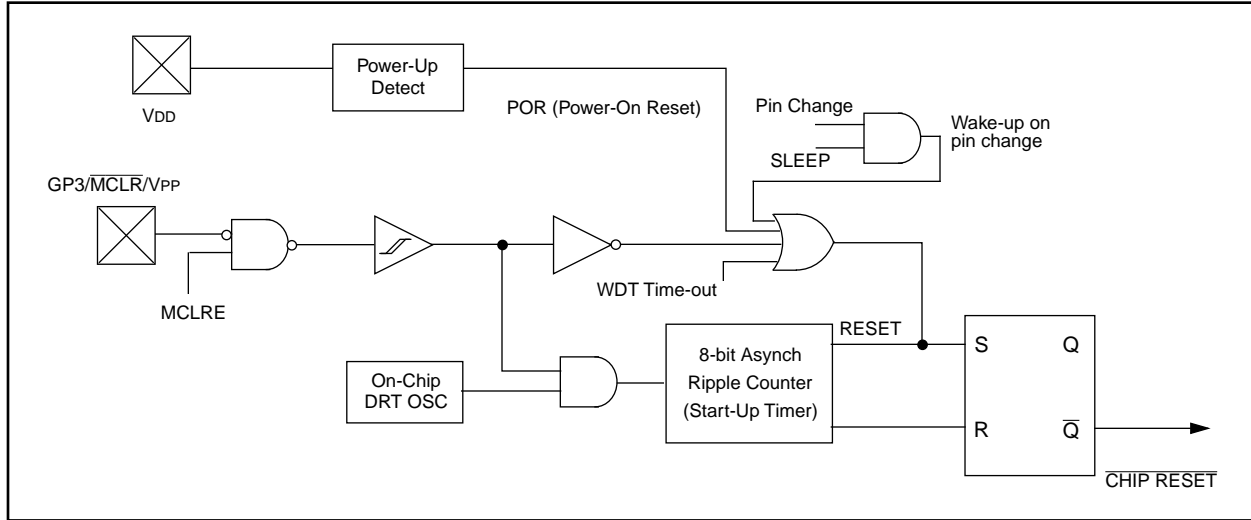
In Figure 7-10, the on-chip Power-On Reset feature is being used (MCLR and VDD are tied together or the pin is programmed to be GP3.). The VDD is stable before the start-up timer times out and there is no problem in getting a proper reset. However, Figure 7-11 depicts a problem situation where VDD rises too slowly. The time between when the DRT senses that MCLR is high and when MCLR (and VDD) actually reach their full value, is too long. In this situation, when the start-up timer times out, VDD has not reached the VDD (min) value and the chip is, therefore, not guaranteed to function correctly. For such situations, we recommend that external RC circuits be used to achieve longer POR delay times (Figure 7-10).

**Note:** When the device starts normal operation (exits the reset condition), device operating parameters (voltage, frequency, temperature, etc.) must be met to ensure operation. If these conditions are not met, the device must be held in reset until the operating conditions are met.

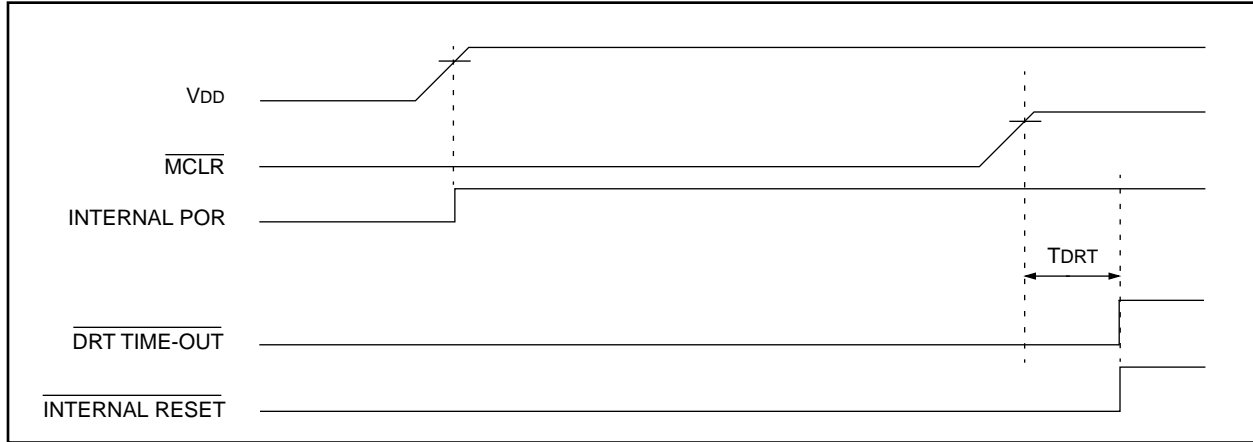
For additional information refer to Application Notes "Power-Up Considerations" - AN522 and "Power-up Trouble Shooting" - AN607.

# PIC12C5XX

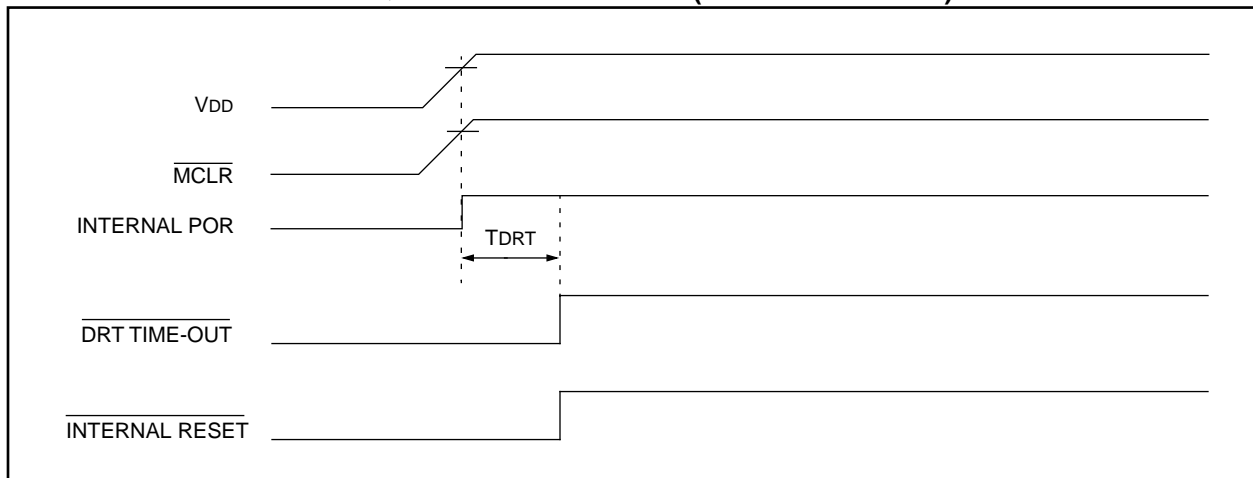
FIGURE 7-8: SIMPLIFIED BLOCK DIAGRAM OF ON-CHIP RESET CIRCUIT



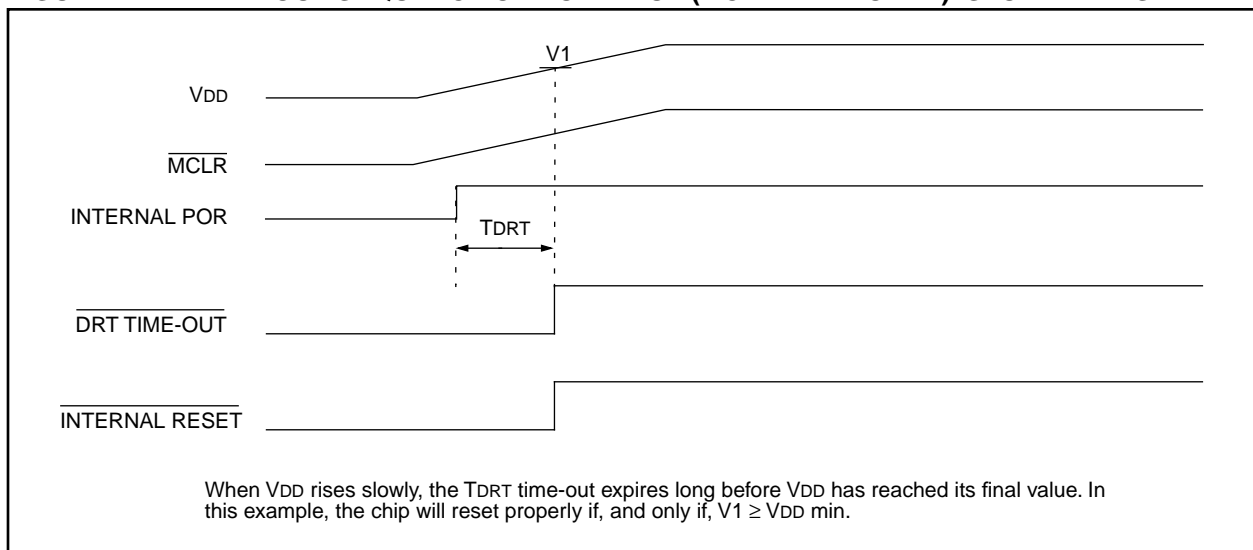
**FIGURE 7-9: TIME-OUT SEQUENCE ON POWER-UP ( $\overline{\text{MCLR}}$  PULLED LOW)**



**FIGURE 7-10: TIME-OUT SEQUENCE ON POWER-UP ( $\overline{\text{MCLR}}$  TIED TO  $V_{DD}$ ): FAST  $V_{DD}$  RISE TIME**



**FIGURE 7-11: TIME-OUT SEQUENCE ON POWER-UP ( $\overline{\text{MCLR}}$  TIED TO  $V_{DD}$ ): SLOW  $V_{DD}$  RISE TIME**



# PIC12C5XX

## 7.5 Device Reset Timer (DRT)

In the PIC12C5XX, the DRT runs any time the device is powered up. DRT runs from RESET and varies based on oscillator selection (see Table 7-5.)

The Device Reset Timer (DRT) provides a fixed 18 ms nominal time-out on reset. The DRT operates on an internal RC oscillator. The processor is kept in RESET as long as the DRT is active. The DRT delay allows VDD to rise above VDD min., and for the oscillator to stabilize.

Oscillator circuits based on crystals or ceramic resonators require a certain time after power-up to establish a stable oscillation. The on-chip DRT keeps the device in a RESET condition for approximately 18 ms after  $\overline{\text{MCLR}}$  has reached a logic high ( $V_{IH\overline{\text{MCLR}}}$ ) level. Thus, programming GP3/ $\overline{\text{MCLR}}$ / $V_{PP}$  as  $\overline{\text{MCLR}}$  and using an external RC network connected to the  $\overline{\text{MCLR}}$  input is not required in most cases, allowing for savings in cost-sensitive and/or space restricted applications, as well as allowing the use of the GP3/ $\overline{\text{MCLR}}$ / $V_{PP}$  pin as a general purpose input.

The Device Reset time delay will vary from chip to chip due to VDD, temperature, and process variation. See AC parameters for details.

The DRT will also be triggered upon a Watchdog Timer time-out (only in XT and LP modes). This is particularly important for applications using the WDT to wake from SLEEP mode automatically.

## 7.6 Watchdog Timer (WDT)

The Watchdog Timer (WDT) is a free running on-chip RC oscillator which does not require any external components. This RC oscillator is separate from the external RC oscillator of the GP5/OSC1/CLKIN pin and the internal 4 MHz oscillator. That means that the WDT will run even if the main processor clock has been stopped, for example, by execution of a SLEEP instruction. During normal operation or SLEEP, a WDT reset or wake-up reset generates a device RESET.

The  $\overline{\text{TO}}$  bit (STATUS<4>) will be cleared upon a Watchdog Timer reset.

The WDT can be permanently disabled by programming the configuration bit WDTE as a '0' (Section 7.1). Refer to the PIC12C5XX Programming Specifications to determine how to access the configuration word.

**TABLE 7-5: DRT (DEVICE RESET TIMER PERIOD)**

Oscillator Configuration	POR Reset	Subsequent Resets
IntRC & ExtRC	18 ms (typical)	300 $\mu$ s (typical)
XT & LP	18 ms (typical)	18 ms (typical)



## 7.6.1 WDT PERIOD

The WDT has a nominal time-out period of 18 ms, (with no prescaler). If a longer time-out period is desired, a prescaler with a division ratio of up to 1:128 can be assigned to the WDT (under software control) by writing to the OPTION register. Thus, a time-out period of a nominal 2.3 seconds can be realized. These periods vary with temperature, VDD and part-to-part process variations (see DC specs).

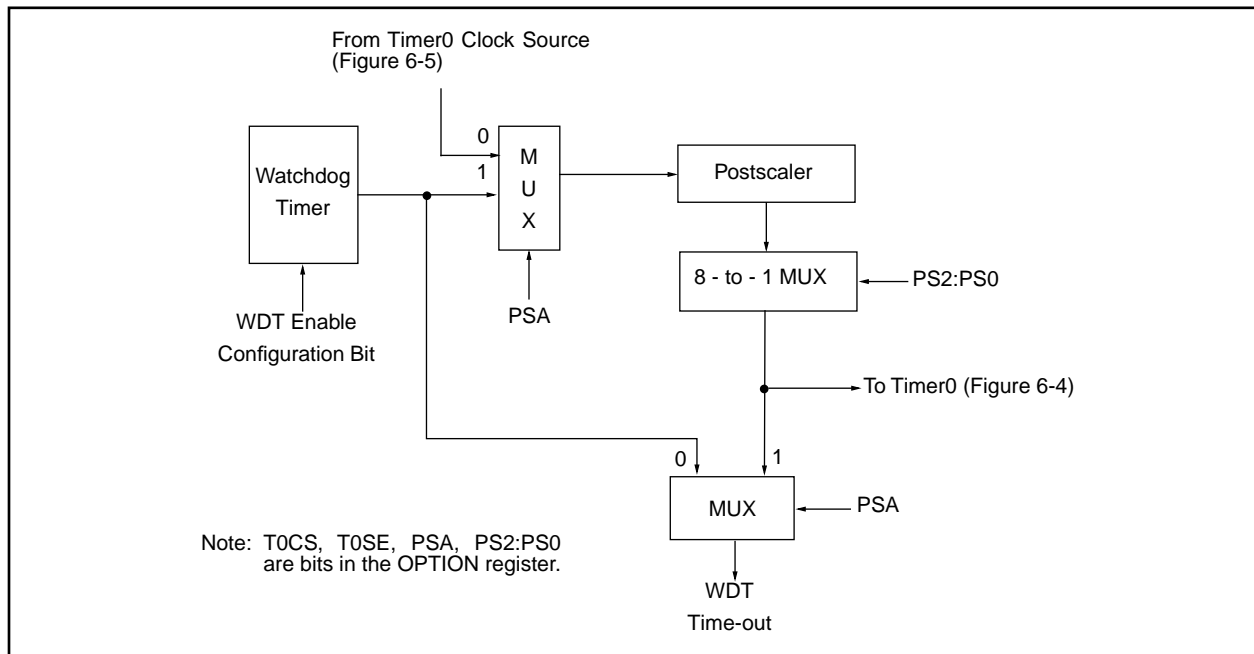
Under worst case conditions (VDD = Min., Temperature = Max., max. WDT prescaler), it may take several seconds before a WDT time-out occurs.

## 7.6.2 WDT PROGRAMMING CONSIDERATIONS

The CLRWDT instruction clears the WDT and the postscaler, if assigned to the WDT, and prevents it from timing out and generating a device RESET.

The SLEEP instruction resets the WDT and the postscaler, if assigned to the WDT. This gives the maximum SLEEP time before a WDT wake-up reset.

**FIGURE 7-12: WATCHDOG TIMER BLOCK DIAGRAM**



**TABLE 7-6: SUMMARY OF REGISTERS 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 MCLR and WDT Reset	Value on Wake-up on Pin Change
N/A	OPTION	GPWU	GPPU	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111	1111 1111

Legend: Shaded boxes = Not used by Watchdog Timer, – = unimplemented, read as '0', u = unchanged

# PIC12C5XX

## 7.7 Time-Out Sequence, Power Down, and Wake-up from SLEEP Status Bits ( $\overline{TO}/\overline{PD}/GPWUF$ )

The  $\overline{TO}$ ,  $\overline{PD}$ , and GPWUF bits in the STATUS register can be tested to determine if a RESET condition has been caused by a power-up condition, a  $\overline{MCLR}$  or Watchdog Timer (WDT) reset, or a  $\overline{MCLR}$  or WDT reset.

**TABLE 7-7:  $\overline{TO}/\overline{PD}/GPWUF$  STATUS AFTER RESET**

GPWUF	$\overline{TO}$	$\overline{PD}$	RESET caused by
0	0	0	WDT wake-up from SLEEP
0	0	1	WDT time-out (not from SLEEP)
0	1	0	$\overline{MCLR}$ wake-up from SLEEP
0	1	1	Power-up
0	u	u	$\overline{MCLR}$ not during SLEEP
1	1	0	Wake-up from SLEEP on pin change

Legend: u = unchanged  
 Note 1: The  $\overline{TO}$ ,  $\overline{PD}$ , and GPWUF bits maintain their status (u) until a reset occurs. A low-pulse on the  $\overline{MCLR}$  input does not change the  $\overline{TO}$ ,  $\overline{PD}$ , and GPWUF status bits.

These STATUS bits are only affected by events listed in Table 7-8.

**TABLE 7-8: EVENTS AFFECTING  $\overline{TO}/\overline{PD}$  STATUS BITS**

Event	GPWUF	$\overline{TO}$	$\overline{PD}$	Remarks
Power-up	0	1	1	
WDT Time-out	0	0	u	No effect on $\overline{PD}$
SLEEP instruction	u	1	0	
CLRWDT instruction	u	1	1	
Wake-up from SLEEP on pin change	1	1	0	

Legend: u = unchanged  
 A WDT time-out will occur regardless of the status of the  $\overline{TO}$  bit. A SLEEP instruction will be executed, regardless of the status of the  $\overline{PD}$  bit. Table 7-7 reflects the status of  $\overline{TO}$  and  $\overline{PD}$  after the corresponding event.

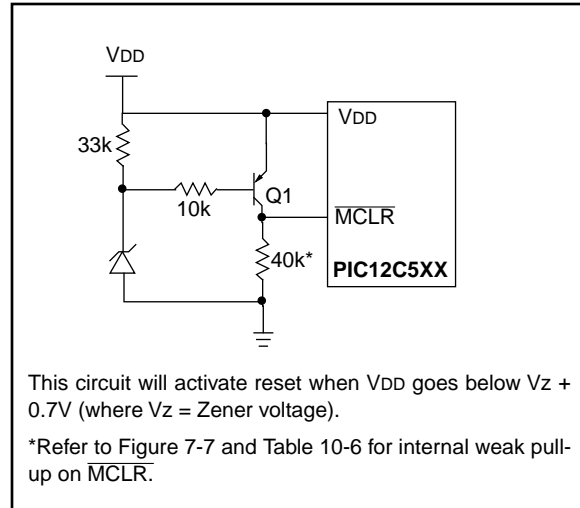
Table 7-4 lists the reset conditions for the special function registers, while Table 7-3 lists the reset conditions for all the registers.

## 7.8 Reset on Brown-Out

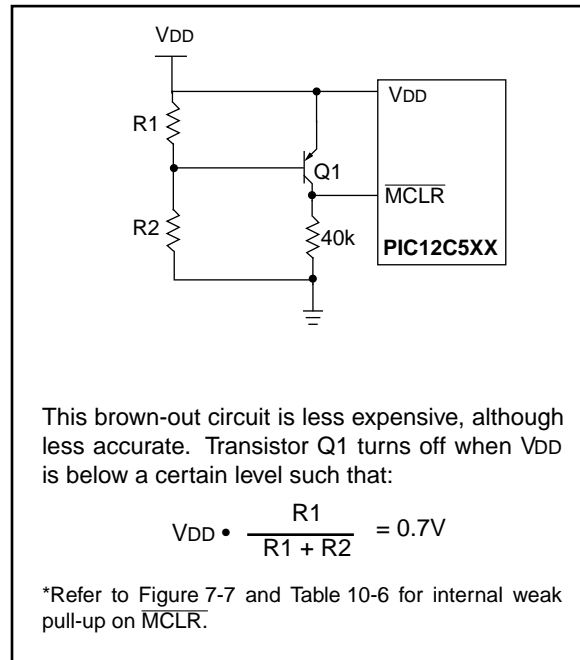
A brown-out is a condition where device power ( $V_{DD}$ ) dips below its minimum value, but not to zero, and then recovers. The device should be reset in the event of a brown-out.

To reset PIC12C5XX devices when a brown-out occurs, external brown-out protection circuits may be built, as shown in Figure 7-13 and Figure 7-14.

**FIGURE 7-13: BROWN-OUT PROTECTION CIRCUIT 1**



**FIGURE 7-14: BROWN-OUT PROTECTION CIRCUIT 2**



## 7.9 Power-Down Mode (SLEEP)

A device may be powered down (SLEEP) and later powered up (Wake-up from SLEEP).

### 7.9.1 SLEEP

The Power-Down mode is entered by executing a SLEEP instruction.

If enabled, the Watchdog Timer will be cleared but keeps running, the  $\overline{TO}$  bit (STATUS<4>) is set, the  $\overline{PD}$  bit (STATUS<3>) is cleared and the oscillator driver is turned off. The I/O ports maintain the status they had before the SLEEP instruction was executed (driving high, driving low, or hi-impedance).

It should be noted that a RESET generated by a WDT time-out does not drive the  $\overline{MCLR}$  pin low.

For lowest current consumption while powered down, the TOCKI input should be at VDD or VSS and the GP3/ $\overline{MCLR}/VPP$  pin must be at a logic high level ( $V_{IHMC}$ ) if  $\overline{MCLR}$  is enabled.

### 7.9.2 WAKE-UP FROM SLEEP

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

1. An external reset input on GP3/ $\overline{MCLR}/VPP$  pin, when configured as  $\overline{MCLR}$ .
2. A Watchdog Timer time-out reset (if WDT was enabled).
3. A change on input pin GP0, GP1, or GP3/ $\overline{MCLR}/VPP$  when wake-up on change is enabled.

These events cause a device reset. The  $\overline{TO}$ ,  $\overline{PD}$ , and GPWUF bits can be used to determine the cause of device reset. The  $\overline{TO}$  bit is cleared if a WDT time-out occurred (and caused wake-up). The  $\overline{PD}$  bit, which is set on power-up, is cleared when SLEEP is invoked. The GPWUF bit indicates a change in state while in SLEEP at pins GP0, GP1, or GP3 (since the last time there was a file or bit operation on GP port).

**Caution:** Right before entering SLEEP, read the input pins. When in SLEEP, wake up occurs when the values at the pins change from the state they were in at the last reading. If a wake-up on change occurs and the pins are not read before reentering SLEEP, a wake up will occur immediately even if no pins change while in SLEEP mode.

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

## 7.10 Program Verification/Code Protection

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

The first 64 locations can be read regardless of the code protection bit setting.

### 7.11 ID Locations

Four memory locations are designated as ID locations where the user can store checksum or other code-identification numbers. These locations are not accessible during normal execution but are readable and writable during program/verify.

Use only the lower 4 bits of the ID locations and always program the upper 8 bits as '0's.

# PIC12C5XX

## 7.12 In-Circuit Serial Programming

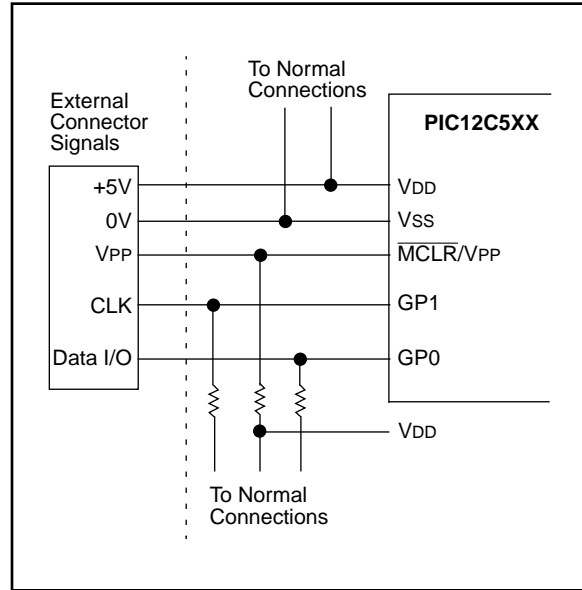
The PIC12C5XX microcontrollers can be serially programmed while in the end application circuit. This is simply done with two lines for clock and data, and three other lines for power, ground, and the programming voltage. This allows customers to manufacture boards with unprogrammed devices, and then program the microcontroller just before shipping the product. This also allows the most recent firmware or a custom firmware to be programmed.

The device is placed into a program/verify mode by holding the GP1 and GP0 pins low while raising the  $\overline{\text{MCLR}}$  ( $V_{PP}$ ) pin from  $V_{IL}$  to  $V_{IH}$  (see programming specification). GP1 becomes the programming clock and GP0 becomes the programming data. Both GP1 and GP0 are Schmitt Trigger inputs in this mode.

After reset, a 6-bit command is then supplied to the device. Depending on the command, 14-bits of program data are then supplied to or from the device, depending if the command was a load or a read. For complete details of serial programming, please refer to the PIC12C5XX Programming Specifications.

A typical in-circuit serial programming connection is shown in Figure 7-15.

**FIGURE 7-15: TYPICAL IN-CIRCUIT SERIAL PROGRAMMING CONNECTION**



## 8.0 INSTRUCTION SET SUMMARY

Each PIC12C5XX instruction is a 12-bit word divided into an OPCODE, which specifies the instruction type, and one or more operands which further specify the operation of the instruction. The PIC12C5XX instruction set summary in Table 8-2 groups the instructions into byte-oriented, bit-oriented, and literal and control operations. Table 8-1 shows the opcode field descriptions.

For **byte-oriented** instructions, 'f' represents a file register designator and 'd' represents a destination designator. The file register designator is used to specify which one of the 32 file registers is to be used by the instruction.

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

For **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.

For **literal and control** operations, 'k' represents an 8 or 9-bit constant or literal value.

**TABLE 8-1: OPCODE FIELD DESCRIPTIONS**

Field	Description
f	Register file address (0x00 to 0x7F)
W	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; d = 0 (store result in W) d = 1 (store result in file register 'f') Default is d = 1
label	Label name
TOS	Top of Stack
PC	Program Counter
WDT	Watchdog Timer Counter
$\overline{TO}$	Time-Out bit
$\overline{PD}$	Power-Down bit
dest	Destination, either the W 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)

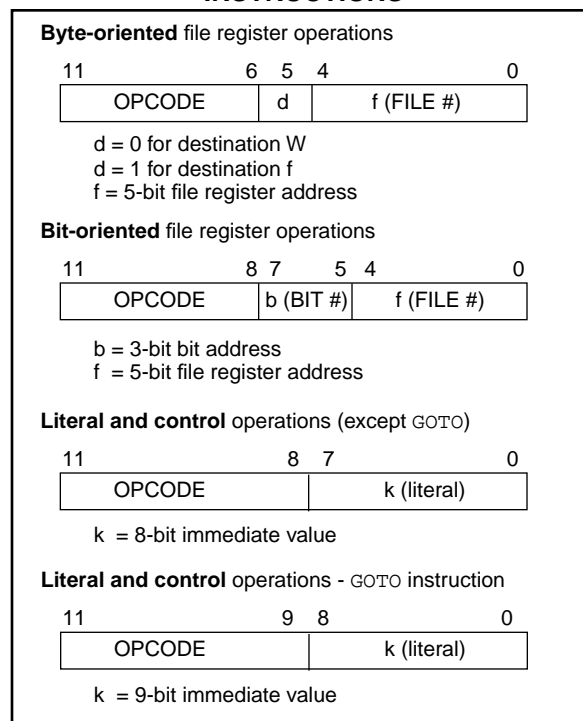
All instructions are executed within a single instruction cycle, unless a conditional test is true or the program counter is changed as a result of an instruction. In this case, the execution takes two instruction cycles. One instruction cycle consists of four oscillator periods. Thus, for an oscillator frequency of 4 MHz, the normal instruction execution time is 1 μs. If a conditional test is true or the program counter is changed as a result of an instruction, the instruction execution time is 2 μs.

Figure 8-1 shows the three general formats that the instructions can have. All examples in the figure use the following format to represent a hexadecimal number:

0xhhh

where 'h' signifies a hexadecimal digit.

**FIGURE 8-1: GENERAL FORMAT FOR INSTRUCTIONS**



# PIC12C5XX

TABLE 8-2: INSTRUCTION SET SUMMARY

Mnemonic, Operands	Description	Cycles	12-Bit Opcode			Status Affected	Notes
			MSb	LSb			
<b>ADDWF</b> f,d	Add W and f	1	0001	11df	ffff	C,DC,Z	1,2,4
<b>ANDWF</b> f,d	AND W with f	1	0001	01df	ffff	Z	2,4
<b>CLRF</b> f	Clear f	1	0000	011f	ffff	Z	4
<b>CLRW</b> –	Clear W	1	0000	0100	0000	Z	
<b>COMF</b> f,d	Complement f	1	0010	01df	ffff	Z	
<b>DECf</b> f,d	Decrement f	1	0000	11df	ffff	Z	2,4
<b>DECFSZ</b> f,d	Decrement f, Skip if 0	1(2)	0010	11df	ffff	None	2,4
<b>INCF</b> f,d	Increment f	1	0010	10df	ffff	Z	2,4
<b>INCFSZ</b> f,d	Increment f, Skip if 0	1(2)	0011	11df	ffff	None	2,4
<b>IORWF</b> f,d	Inclusive OR W with f	1	0001	00df	ffff	Z	2,4
<b>MOVF</b> f,d	Move f	1	0010	00df	ffff	Z	2,4
<b>MOVWF</b> f	Move W to f	1	0000	001f	ffff	None	1,4
<b>NOP</b> –	No Operation	1	0000	0000	0000	None	
<b>RLF</b> f,d	Rotate left f through Carry	1	0011	01df	ffff	C	2,4
<b>RRF</b> f,d	Rotate right f through Carry	1	0011	00df	ffff	C	2,4
<b>SUBWF</b> f,d	Subtract W from f	1	0000	10df	ffff	C,DC,Z	1,2,4
<b>SWAPF</b> f,d	Swap f	1	0011	10df	ffff	None	2,4
<b>XORWF</b> f,d	Exclusive OR W with f	1	0001	10df	ffff	Z	2,4
<b>BIT-ORIENTED FILE REGISTER OPERATIONS</b>							
<b>BCF</b> f,b	Bit Clear f	1	0100	bbbf	ffff	None	2,4
<b>BSF</b> f,b	Bit Set f	1	0101	bbbf	ffff	None	2,4
<b>BTFSC</b> f,b	Bit Test f, Skip if Clear	1 (2)	0110	bbbf	ffff	None	
<b>BTFSS</b> f,b	Bit Test f, Skip if Set	1 (2)	0111	bbbf	ffff	None	
<b>LITERAL AND CONTROL OPERATIONS</b>							
<b>ANDLW</b> k	AND literal with W	1	1110	kkkk	kkkk	Z	
<b>CALL</b> k	Call subroutine	2	1001	kkkk	kkkk	None	1
<b>CLRWDT</b> k	Clear Watchdog Timer	1	0000	0000	0100	$\overline{TO}$ , PD	
<b>GOTO</b> k	Unconditional branch	2	101k	kkkk	kkkk	None	
<b>IORLW</b> k	Inclusive OR Literal with W	1	1101	kkkk	kkkk	Z	
<b>MOVLW</b> k	Move Literal to W	1	1100	kkkk	kkkk	None	
<b>OPTION</b> –	Load OPTION register	1	0000	0000	0010	None	
<b>RETLW</b> k	Return, place Literal in W	2	1000	kkkk	kkkk	None	
<b>SLEEP</b> –	Go into standby mode	1	0000	0000	0011	$\overline{TO}$ , PD	
<b>TRIS</b> f	Load TRIS register	1	0000	0000	0fff	None	3
<b>XORLW</b> k	Exclusive OR Literal to W	1	1111	kkkk	kkkk	Z	

Note 1: The 9th bit of the program counter will be forced to a '0' by any instruction that writes to the PC except for **GOTO**. (Section 4.5)

- When an I/O register is modified as a function of itself (e.g. **MOVF** GPIO, 1), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.
- The instruction **TRIS** f, where f = 6 causes the contents of the W register to be written to the tristate latches of GPIO. A '1' forces the pin to a hi-impedance state and disables the output buffers.
- If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared (if assigned to TMR0).

## **ADDWF**      **Add W and f**

Syntax:      [ *label* ] ADDWF   f,d

Operands:     $0 \leq f \leq 31$   
 $d \in [0,1]$

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

Status Affected: C, DC, Z

Encoding:    

0001	11df	ffff
------	------	------

Description:    Add the contents of the W register and register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is '1' the result is stored back in register 'f'.

Words:        1

Cycles:        1

Example:      ADDWF   FSR, 0

Before Instruction

W = 0x17  
 FSR = 0xC2

After Instruction

W = 0xD9  
 FSR = 0xC2

## **ANDWF**      **AND W with f**

Syntax:      [ *label* ] ANDWF   f,d

Operands:     $0 \leq f \leq 31$   
 $d \in [0,1]$

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

Status Affected: Z

Encoding:    

0001	01df	ffff
------	------	------

Description:    The contents of the W register are AND'ed with register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is '1' the result is stored back in register 'f'.

Words:        1

Cycles:        1

Example:      ANDWF   FSR, 1

Before Instruction

W = 0x17  
 FSR = 0xC2

After Instruction

W = 0x17  
 FSR = 0x02

## **ANDLW**      **And literal with W**

Syntax:      [ *label* ] ANDLW   k

Operands:     $0 \leq k \leq 255$

Operation:     $(W) .\text{AND.} (k) \rightarrow (W)$

Status Affected: Z

Encoding:    

1110	kkkk	kkkk
------	------	------

Description:    The contents of the W register are AND'ed with the eight-bit literal 'k'. The result is placed in the W register.

Words:        1

Cycles:        1

Example:      ANDLW   0x5F

Before Instruction

W = 0xA3

After Instruction

W = 0x03

## **BCF**          **Bit Clear f**

Syntax:      [ *label* ] BCF    f,b

Operands:     $0 \leq f \leq 31$   
 $0 \leq b \leq 7$

Operation:     $0 \rightarrow (f<b>)$

Status Affected: None

Encoding:    

0100	bbbf	ffff
------	------	------

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

Words:        1

Cycles:        1

Example:      BCF      FLAG\_REG, 7

Before Instruction

FLAG\_REG = 0xC7

After Instruction

FLAG\_REG = 0x47

# PIC12C5XX

## BSF Bit Set f

Syntax: [ *label* ] BSF f,b  
 Operands:  $0 \leq f \leq 31$   
 $0 \leq b \leq 7$   
 Operation:  $1 \rightarrow (f<b>)$   
 Status Affected: None  
 Encoding: 

0101	bbbbf	ffff
------	-------	------

  
 Description: Bit 'b' in register 'f' is set.  
 Words: 1  
 Cycles: 1  
 Example: BSF FLAG\_REG, 7

Before Instruction  
 FLAG\_REG = 0x0A  
 After Instruction  
 FLAG\_REG = 0x8A

## BTFSC Bit Test f, Skip if Clear

Syntax: [ *label* ] BTFSC f,b  
 Operands:  $0 \leq f \leq 31$   
 $0 \leq b \leq 7$   
 Operation: skip if (f<b>) = 0  
 Status Affected: None  
 Encoding: 

0110	bbbbf	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 an NOP is executed instead, making this a 2 cycle instruction.  
 Words: 1  
 Cycles: 1(2)  
 Example: HERE BTFSC FLAG, 1  
 FALSE GOTO PROCESS\_CODE  
 TRUE •  
 •  
 •

Before Instruction  
 PC = address (HERE)  
 After Instruction  
 if FLAG<1> = 0,  
 PC = address (TRUE);  
 if FLAG<1> = 1,  
 PC = address (FALSE)

## BTFSS Bit Test f, Skip if Set

Syntax: [ *label* ] BTFSS f,b  
 Operands:  $0 \leq f \leq 31$   
 $0 \leq b < 7$   
 Operation: skip if (f<b>) = 1  
 Status Affected: None  
 Encoding: 

0111	bbbbf	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 2 cycle instruction.  
 Words: 1  
 Cycles: 1(2)  
 Example: HERE BTFSS FLAG, 1  
 FALSE GOTO PROCESS\_CODE  
 TRUE •  
 •  
 •

Before Instruction  
 PC = address (HERE)  
 After Instruction  
 if FLAG<1> = 0,  
 PC = address (FALSE);  
 if FLAG<1> = 1,  
 PC = address (TRUE)



## CALL Subroutine Call

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

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

**Operation:** (PC) + 1 → Top of Stack;  
 $k \rightarrow PC<7:0>$ ;  
 (STATUS<6:5>) → PC<10:9>;  
 $0 \rightarrow PC<8>$

**Status Affected:** None

**Encoding:**

1001	kkkk	kkkk
------	------	------

**Description:** Subroutine call. First, return address (PC+1) is pushed onto the stack. The eight bit immediate address is loaded into PC bits <7:0>. The upper bits PC<10:9> are loaded from STATUS<6:5>, PC<8> is cleared. CALL is a two cycle instruction.

**Words:** 1

**Cycles:** 2

**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*

**Operands:**  $0 \leq f \leq 31$

**Operation:**  $00h \rightarrow (f)$ ;  
 $1 \rightarrow Z$

**Status Affected:** Z

**Encoding:**

0000	011f	ffff
------	------	------

**Description:** The contents of register 'f' are cleared and the Z bit is set.

**Words:** 1

**Cycles:** 1

**Example:**     CLRF     FLAG\_REG

Before Instruction  
 FLAG\_REG = 0x5A

After Instruction  
 FLAG\_REG = 0x00  
 Z = 1

## CLRW Clear W

**Syntax:** [ *label* ] CLRW

**Operands:** None

**Operation:**  $00h \rightarrow (W)$ ;  
 $1 \rightarrow Z$

**Status Affected:** Z

**Encoding:**

0000	0100	0000
------	------	------

**Description:** The W register is cleared. Zero bit (Z) is set.

**Words:** 1

**Cycles:** 1

**Example:**     CLRW

Before Instruction  
 W = 0x5A

After Instruction  
 W = 0x00  
 Z = 1

## CLRWDT Clear Watchdog Timer

**Syntax:** [ *label* ] CLRWDT

**Operands:** None

**Operation:**  $00h \rightarrow WDT$ ;  
 $0 \rightarrow WDT$  prescaler (if assigned);  
 $1 \rightarrow \overline{TO}$ ;  
 $1 \rightarrow \overline{PD}$

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

**Encoding:**

0000	0000	0100
------	------	------

**Description:** The CLRWDT instruction resets the WDT. It also resets the prescaler, if the prescaler is assigned to the WDT and not Timer0. Status bits  $\overline{TO}$  and  $\overline{PD}$  are set.

**Words:** 1

**Cycles:** 1

**Example:**     CLRWDT

Before Instruction  
 WDT counter = ?

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

# PIC12C5XX

## COMF      Complement f

**Syntax:**      [ *label* ] COMF f,d

**Operands:**     $0 \leq f \leq 31$   
                    $d \in [0,1]$

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

**Status Affected:** Z

**Encoding:**

0010	01df	ffff
------	------	------

**Description:**    The contents of register 'f' are complemented. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.

**Words:**        1

**Cycles:**        1

**Example:**        COMF    REG1, 0

```

Before Instruction
  REG1 = 0x13

After Instruction
  REG1 = 0x13
  W     = 0xEC
  
```

## DECF      Decrement f

**Syntax:**      [ *label* ] DECF f,d

**Operands:**     $0 \leq f \leq 31$   
                    $d \in [0,1]$

**Operation:**     $(f) - 1 \rightarrow (\text{dest})$

**Status Affected:** Z

**Encoding:**

0000	11df	ffff
------	------	------

**Description:**    Decrement register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.

**Words:**        1

**Cycles:**        1

**Example:**        DECF    CNT, 1

```

Before Instruction
  CNT = 0x01
  Z   = 0

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

## DECFSZ    Decrement f, Skip if 0

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

**Operands:**     $0 \leq f \leq 31$   
                    $d \in [0,1]$

**Operation:**     $(f) - 1 \rightarrow d$ ; skip if result = 0

**Status Affected:** None

**Encoding:**

0010	11df	ffff
------	------	------

**Description:**    The contents of register 'f' are decremented. If 'd' is 0 the result is placed in the W register. 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)

**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 ≠ 0,
  PC = address (HERE+1)
  
```

## GOTO      Unconditional Branch

**Syntax:**      [ *label* ] GOTO k

**Operands:**     $0 \leq k \leq 511$

**Operation:**     $k \rightarrow PC\langle 8:0 \rangle$ ;  
                    $STATUS\langle 6:5 \rangle \rightarrow PC\langle 10:9 \rangle$

**Status Affected:** None

**Encoding:**

101k	kkkk	kkkk
------	------	------

**Description:**    GOTO is an unconditional branch. The 9-bit immediate value is loaded into PC bits  $\langle 8:0 \rangle$ . The upper bits of PC are loaded from  $STATUS\langle 6:5 \rangle$ . GOTO is a two cycle instruction.

**Words:**        1

**Cycles:**        2

**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 ≤ 31 d ∈ [0,1]			
Operation:	(f) + 1 → (dest)			
Status Affected:	Z			
Encoding:	<table border="1"><tr><td>0010</td><td>10df</td><td>ffff</td></tr></table>	0010	10df	ffff
0010	10df	ffff		
Description:	The contents of register 'f' are incremented. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.			
Words:	1			
Cycles:	1			
Example:	INCF CNT, 1			
	Before Instruction			
	CNT = 0xFF			
	Z = 0			
	After Instruction			
	CNT = 0x00			
	Z = 1			

<b>INCFSZ</b>	<b>Increment f, Skip if 0</b>			
Syntax:	[ <i>label</i> ] INCFSZ f,d			
Operands:	0 ≤ f ≤ 31 d ∈ [0,1]			
Operation:	(f) + 1 → (dest), skip if result = 0			
Status Affected:	None			
Encoding:	<table border="1"><tr><td>0011</td><td>11df</td><td>ffff</td></tr></table>	0011	11df	ffff
0011	11df	ffff		
Description:	The contents of register 'f' are incremented. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'. If the result is 0, then 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)			
Example:	HERE INCFSZ CNT, 1 GOTO LOOP CONTINUE • • •			
	Before Instruction			
	PC = address (HERE)			
	After Instruction			
	CNT = CNT + 1;			
	if CNT = 0,			
	PC = address (CONTINUE);			
	if CNT ≠ 0,			
	PC = address (HERE +1)			

<b>IORLW</b>	<b>Inclusive OR literal with W</b>			
Syntax:	[ <i>label</i> ] IORLW k			
Operands:	0 ≤ k ≤ 255			
Operation:	(W) .OR. (k) → (W)			
Status Affected:	Z			
Encoding:	<table border="1"><tr><td>1101</td><td>kkkk</td><td>kkkk</td></tr></table>	1101	kkkk	kkkk
1101	kkkk	kkkk		
Description:	The contents of the W register are OR'ed with the eight bit literal 'k'. The result is placed in the W register.			
Words:	1			
Cycles:	1			
Example:	IORLW 0x35			
	Before Instruction			
	W = 0x9A			
	After Instruction			
	W = 0xBF			
	Z = 0			

<b>IORWF</b>	<b>Inclusive OR W with f</b>			
Syntax:	[ <i>label</i> ] IORWF f,d			
Operands:	0 ≤ f ≤ 31 d ∈ [0,1]			
Operation:	(W).OR. (f) → (dest)			
Status Affected:	Z			
Encoding:	<table border="1"><tr><td>0001</td><td>00df</td><td>ffff</td></tr></table>	0001	00df	ffff
0001	00df	ffff		
Description:	Inclusive OR the W register with register 'f'. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.			
Words:	1			
Cycles:	1			
Example:	IORWF RESULT, 0			
	Before Instruction			
	RESULT = 0x13			
	W = 0x91			
	After Instruction			
	RESULT = 0x13			
	W = 0x93			
	Z = 0			

# PIC12C5XX

**MOVF**      **Move f**

---

Syntax:      [ *label* ] MOVF f,d

Operands:     $0 \leq f \leq 31$   
               $d \in [0,1]$

Operation:    (f) → (dest)

Status Affected: Z

Encoding:    

0010	00df	ffff
------	------	------

Description:    The contents of register 'f' is moved to destination 'd'. If 'd' is 0, destination is the W register. If 'd' is 1, the destination is file register 'f'. 'd' is 1 is useful to test a file register since status flag Z is affected.

Words:        1

Cycles:       1

Example:      MOVF    FSR,    0

              After Instruction  
              W    =    value in FSR register

**MOVLW**     **Move Literal to W**

---

Syntax:      [ *label* ] MOVLW k

Operands:     $0 \leq k \leq 255$

Operation:     $k \rightarrow (W)$

Status Affected: None

Encoding:    

1100	kkkk	kkkk
------	------	------

Description:    The eight bit literal 'k' is loaded into the W register. The don't cares will assemble as 0s.

Words:        1

Cycles:       1

Example:      MOVLW   0x5A

              After Instruction  
              W    =    0x5A

**MOVWF**      **Move W to f**

---

Syntax:      [ *label* ] MOVWF f

Operands:     $0 \leq f \leq 31$

Operation:    (W) → (f)

Status Affected: None

Encoding:    

0000	001f	ffff
------	------	------

Description:    Move data from the W register to register 'f'.

Words:        1

Cycles:       1

Example:      MOVWF    TEMP\_REG

              Before Instruction  
              TEMP\_REG = 0xFF  
              W        = 0x4F

              After Instruction  
              TEMP\_REG = 0x4F  
              W        = 0x4F

**NOP**        **No Operation**

---

Syntax:      [ *label* ] NOP

Operands:    None

Operation:    No operation

Status Affected: None

Encoding:    

0000	0000	0000
------	------	------

Description:    No operation.

Words:        1

Cycles:       1

Example:      NOP

## OPTION Load OPTION Register

Syntax: [ *label* ] OPTION  
 Operands: None  
 Operation: (W) → OPTION  
 Status Affected: None  
 Encoding: 

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

  
 Description: The content of the W register is loaded into the OPTION register.  
 Words: 1  
 Cycles: 1  
 Example           OPTION

Before Instruction  
 W = 0x07  
 After Instruction  
 OPTION = 0x07

## RETLW Return with Literal in W

Syntax: [ *label* ] RETLW k  
 Operands:  $0 \leq k \leq 255$   
 Operation:  $k \rightarrow (W)$ ;  
 TOS → PC  
 Status Affected: None  
 Encoding: 

1000	kkkk	kkkk
------	------	------

Description: The W register is loaded with the eight bit literal 'k'. The program counter is loaded from the top of the stack (the return address). This is a two cycle instruction.

Words: 1  
 Cycles: 2

Example: 

```
CALL TABLE ;W contains
           ;table offset
           ;value.
           ;W now has table
           ;value.
           .
TABLE ADDWF PC ;W = offset
      RETLW k1 ;Begin table
      RETLW k2 ;
      .
      .
      .
      RETLW kn ; End of table
```

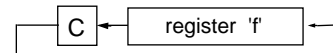
Before Instruction  
 W = 0x07  
 After Instruction  
 W = value of k8

## RLF Rotate Left f through Carry

Syntax: [ *label* ] RLF f,d  
 Operands:  $0 \leq f \leq 31$   
 $d \in [0,1]$   
 Operation: See description below  
 Status Affected: C  
 Encoding: 

0011	01df	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 the W register. If 'd' is 1 the result is stored back in register 'f'.



Words: 1  
 Cycles: 1  
 Example: RLF REG1,0

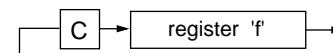
Before Instruction  
 REG1 = 1110 0110  
 C = 0  
 After Instruction  
 REG1 = 1110 0110  
 W = 1100 1100  
 C = 1

## RRF Rotate Right f through Carry

Syntax: [ *label* ] RRF f,d  
 Operands:  $0 \leq f \leq 31$   
 $d \in [0,1]$   
 Operation: See description below  
 Status Affected: C  
 Encoding: 

0011	00df	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 the W register. If 'd' is 1 the result is placed back in register 'f'.



Words: 1  
 Cycles: 1  
 Example: RRF REG1,0

Before Instruction  
 REG1 = 1110 0110  
 C = 0  
 After Instruction  
 REG1 = 1110 0110  
 W = 0111 0011  
 C = 0

# PIC12C5XX

<b>SLEEP</b>	<b>Enter SLEEP Mode</b>			
Syntax:	[ <i>label</i> ] SLEEP			
Operands:	None			
Operation:	00h → WDT; 0 → WDT prescaler; 1 → $\overline{TO}$ ; 0 → $\overline{PD}$			
Status Affected:	$\overline{TO}$ , $\overline{PD}$ , GPWUF			
Encoding:	<table border="1"><tr><td>0000</td><td>0000</td><td>0011</td></tr></table>	0000	0000	0011
0000	0000	0011		
Description:	Time-out status bit ( $\overline{TO}$ ) is set. The power down status bit ( $\overline{PD}$ ) is cleared. GPWUF is unaffected. The WDT and its prescaler are cleared. The processor is put into SLEEP mode with the oscillator stopped. See section on SLEEP for more details.			
Words:	1			
Cycles:	1			
Example:	SLEEP			

<b>SUBWF</b>	<b>Subtract W from f</b>			
Syntax:	[ <i>label</i> ] SUBWF f,d			
Operands:	$0 \leq f \leq 31$ $d \in [0,1]$			
Operation:	$(f) - (W) \rightarrow (\text{dest})$			
Status Affected:	C, DC, Z			
Encoding:	<table border="1"><tr><td>0000</td><td>10df</td><td>ffff</td></tr></table>	0000	10df	ffff
0000	10df	ffff		
Description:	Subtract (2's complement method) the W register from register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.			
Words:	1			
Cycles:	1			
<u>Example 1:</u>	SUBWF REG1, 1			
Before Instruction	REG1 = 3 W = 2 C = ?			
After Instruction	REG1 = 1 W = 2 C = 1 ; result is positive			

<u>Example 2:</u>	
Before Instruction	REG1 = 2 W = 2 C = ?
After Instruction	REG1 = 0 W = 2 C = 1 ; result is zero

<u>Example 3:</u>	
Before Instruction	REG1 = 1 W = 2 C = ?
After Instruction	REG1 = FF W = 2 C = 0 ; result is negative

## SWAPF      Swap Nibbles in f

Syntax:      `[label] SWAPF f,d`

Operands:     $0 \leq f \leq 31$   
 $d \in [0,1]$

Operation:     $(f<3:0>) \rightarrow (dest<7:4>);$   
 $(f<7:4>) \rightarrow (dest<3:0>)$

Status Affected: None

Encoding:    

0011	10df	ffff
------	------	------

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

Words:      1

Cycles:      1

Example      `SWAPF REG1, 0`

Before Instruction  
REG1 = 0xA5

After Instruction  
REG1 = 0xA5  
W = 0x5A

## TRIS      Load TRIS Register

Syntax:      `[label] TRIS f`

Operands:     $f = 6$

Operation:     $(W) \rightarrow \text{TRIS register } f$

Status Affected: None

Encoding:    

0000	0000	0fff
------	------	------

Description:    TRIS register 'f' ( $f = 6$ ) is loaded with the contents of the W register

Words:      1

Cycles:      1

Example      `TRIS GPIO`

Before Instruction  
W = 0xA5

After Instruction  
TRIS = 0xA5

**Note:**  $f = 6$  for PIC12C5XX only.

## XORLW      Exclusive OR literal with W

Syntax:      `[label] XORLW k`

Operands:     $0 \leq k \leq 255$

Operation:     $(W) .\text{XOR. } k \rightarrow (W)$

Status Affected: Z

Encoding:    

1111	kkkk	kkkk
------	------	------

Description:    The contents of the W register are XOR'ed with the eight bit literal 'k'. The result is placed in the W register.

Words:      1

Cycles:      1

Example:      `XORLW 0xAF`

Before Instruction  
W = 0xB5

After Instruction  
W = 0x1A

## XORWF      Exclusive OR W with f

Syntax:      `[label] XORWF f,d`

Operands:     $0 \leq f \leq 31$   
 $d \in [0,1]$

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

Status Affected: Z

Encoding:    

0001	10df	ffff
------	------	------

Description:    Exclusive OR the contents of the W register with register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.

Words:      1

Cycles:      1

Example      `XORWF REG, 1`

Before Instruction  
REG = 0xAF  
W = 0xB5

After Instruction  
REG = 0x1A  
W = 0xB5

# PIC12C5XX

---

NOTES:



## 9.0 DEVELOPMENT SUPPORT

### 9.1 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<sup>®</sup> II Universal Programmer
- PICSTART<sup>®</sup> 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<sup>®</sup>*-MP)

### 9.2 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, PIC14C000, PIC16C5X, PIC16CXXX and PIC17CXX families. PICMASTER is supplied with the MPLAB<sup>™</sup> 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<sup>®</sup> 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.

### 9.3 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<sup>®</sup> through Pentium<sup>™</sup> based machines under Windows 3.x environment. ICEPIC features real time, non-intrusive emulation.

### 9.4 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, PIC14000 and PIC12C5XX devices. It can also set configuration and code-protect bits in this mode.

### 9.5 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.

## 9.6 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.

## 9.7 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.

## 9.8 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.

## 9.9 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.

## 9.10 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 allows full symbolic debugging from PICMASTER, Microchip's Universal Emulator System.

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.

## 9.11 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.

## 9.12 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).

## 9.13 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™* demonstration board for hands-on experience with fuzzy logic systems implementation.

## 9.14 MP-DriveWay™ – 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.

## 9.15 SEEVAL® 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 Serials™ and secure serials. The Total Endurance™ 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.

## 9.16 KEELOQ® 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.

# PIC12C5XX

TABLE 9-1: DEVELOPMENT TOOLS FROM MICROCHIP

	PIC12C5XX	PIC14000	PIC16C5X	PIC16CXXX	PIC16C6X	PIC16C7XX	PIC16C8X	PIC16C9XX	PIC17C4X	PIC17C75X	24CXX 25CXX 93CXX	HCS200 HCS300 HCS301
Emulator Products	PICMASTER® / PICMASTER-CE In-Circuit Emulator	✓	✓	✓	✓	✓	✓	✓	✓	Available 3097		
	ICEPIC Low-Cost In-Circuit Emulator	✓	✓	✓	✓	✓	✓		✓			
Software Tools	MPLAB™ Integrated Development Environment	✓	✓	✓	✓	✓	✓	✓	✓	✓		
	MPLAB™ C Compiler	✓	✓	✓	✓	✓	✓	✓	✓	✓		
	fuzzyTECH®-MP Explorer/Edition Fuzzy Logic Dev. Tool	✓	✓	✓	✓	✓	✓	✓	✓			
	MP-DriveWay™ Applications Code Generator	✓	✓	✓	✓	✓	✓	✓	✓		✓	
Programmers	Total Endurance™ Software Model										✓	
	PICSTART® Lite Ultra Low-Cost Dev. Kit		✓			✓	✓					
Demo Boards	PICSTART® Plus Low-Cost Universal Dev. Kit	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	PRO MATE® II Universal Programmer	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	KEELOQ® Programmer											
	SEEVAL® Designers Kit										✓	
	PICDEM-1		✓	✓		✓			✓			
	PICDEM-2				✓							
	PICDEM-3				✓							
	KEELOQ® Evaluation Kit							✓				✓

## 10.0 ELECTRICAL CHARACTERISTICS - PIC12C5XX

### Absolute Maximum Ratings†

Ambient Temperature under bias .....	-40°C to +125°C
Storage Temperature .....	-65°C to +150°C
Voltage on VDD with respect to VSS .....	0 to +7.5 V
Voltage on $\overline{MCLR}$ with respect to VSS .....	0 to +14 V
Voltage on all other pins with respect to VSS .....	-0.6 V to (VDD + 0.6 V)
Total Power Dissipation <sup>(1)</sup> .....	700 mW
Max. Current out of VSS pin.....	200 mA
Max. Current into VDD pin.....	150 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
Max. Output Current sunk by any I/O pin .....	25 mA
Max. Output Current sourced by any I/O pin.....	25 mA
Max. Output Current sourced by I/O port (GPIO).....	100 mA
Max. Output Current sunk by I/O port (GPIO).....	100 mA

**Note 1:** Power Dissipation is calculated as follows:  $P_{DIS} = V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD} - V_{OH}) \times I_{OH}\} + \sum (V_{OL} \times I_{OL})$

†NOTICE: Stresses above those listed under "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.

# PIC12C5XX

## 10.1 DC CHARACTERISTICS: PIC12508/509 (Commercial) PIC12508/509 (Industrial) PIC12508/509 (Extended)

DC Characteristics Power Supply Pins		Standard Operating Conditions (unless otherwise specified)					
		Operating Temperature 0°C ≤ TA ≤ +70°C (commercial) -40°C ≤ TA ≤ +85°C (industrial) -40°C ≤ TA ≤ +125°C (extended)					
Characteristic	Sym	Min	Typ <sup>(1)</sup>	Max	Units	Conditions	
Supply Voltage	VDD	2.5		5.5	V	FOSC = DC to 4 MHz (Commercial/ Industrial)	
		3.0		5.5	V	FOSC = DC to 4 MHz (Extended)	
RAM Data Retention Voltage <sup>(2)</sup>	VDR		1.5*		V	Device in SLEEP mode	
VDD Start Voltage to ensure Power-on Reset	VPOR		VSS		V	See section on Power-on Reset for details	
VDD Rise Rate to ensure Power-on Reset	SVDD	0.05*			V/ms	See section on Power-on Reset for details	
Supply Current <sup>(3)</sup>	IDD	—	1.8	2.4	mA	XT and EXTRC options (Note 4) FOSC = 4 MHz, VDD = 5.5V	
		—	1.8	2.4	mA	INTRC Option FOSC = 4 MHz, VDD = 5.5V	
		—	15	27	μA	LP OPTION, Commercial Temperature FOSC = 32 kHz, VDD = 3.0V, WDT disabled	
		—	19	35	μA	LP OPTION, Industrial Temperature FOSC = 32 kHz, VDD = 3.0V, WDT disabled	
		—	19	35	μA	LP OPTION, Extended Temperature FOSC = 32 kHz, VDD = 3.0V, WDT disabled	
Power-Down Current <sup>(5)</sup> WDT Enabled	IPD	—	4	12	μA	VDD = 3.0V, Commercial	
		—	4	14	μA	VDD = 3.0V, Industrial	
		—	5	22	μA	VDD = 3.0V, Extended	
		WDT Disabled	—	0.25	4	μA	VDD = 3.0V, Commercial
		—	0.25	5	μA	VDD = 3.0V, Industrial	
		—	2	18	μA	VDD = 3.0V, Extended	

\* These parameters are characterized but not tested.

Note 1: Data in the Typical ("Typ") column is based on characterization results at 25°C. This data is for design guidance only and is not tested.

2: This is the limit to which VDD can be lowered in SLEEP mode without losing RAM data.

3: The supply current is mainly a function of the operating voltage and frequency. Other factors such as bus loading, oscillator type, bus rate, internal code execution pattern, and temperature also have an impact on the current consumption.

a) 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 VSS, T0CKI = VDD, MCLR = VDD; WDT enabled/disabled as specified.

b) For standby current measurements, the conditions are the same, except that the device is in SLEEP mode.

4: Does not include current through Rext. The current through the resistor can be estimated by the formula:  $I_R = V_{DD}/2R_{ext}$  (mA) with Rext in kOhm.

5: 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.

**10.2 DC CHARACTERISTICS:**      **PIC12508/509 (Commercial)**  
**PIC12508/509 (Industrial)**  
**PIC12508/509 (Extended)**

DC Characteristics All Pins Except Power Supply Pins		Standard Operating Conditions (unless otherwise specified)				
		Operating Temperature $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$ (commercial) $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ (industrial) $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ (extended) Operating Voltage $V_{DD}$ range is described in Section 10.1.				
Characteristic	Sym	Min	Typ <sup>(1)</sup>	Max	Units	Conditions
<b>Input Low Voltage</b> I/O ports	V <sub>IL</sub>	V <sub>SS</sub>		0.8	V	Pin at hi-impedance $4.5\text{V} < V_{DD} \leq 5.5\text{V}$
		V <sub>SS</sub>		$0.15 V_{DD}$	V	Pin at hi-impedance $3.0\text{V} < V_{DD} \leq 4.5\text{V}$
		V <sub>SS</sub>		$0.15 V_{DD}$	V	
		V <sub>SS</sub>		$0.3 V_{DD}$	V	
$\overline{\text{MCLR}}$ and GP2 (Schmitt Trigger)		V <sub>SS</sub>		$0.15 V_{DD}$	V	
OSC1		V <sub>SS</sub>		$0.15 V_{DD}$	V	EXTRC option only <sup>(4)</sup>
OSC1		V <sub>SS</sub>		$0.3 V_{DD}$	V	XT and LP options
<b>Input High Voltage</b> I/O ports	V <sub>IH</sub>	$0.25V_{DD}+0.8\text{V}$		V <sub>DD</sub>	V	$2.5\text{V} < V_{DD} \leq 4.5\text{V}$
		2.0		V <sub>DD</sub>	V	$4.5\text{V} < V_{DD} \leq 5.5\text{V}$ <sup>(5)</sup>
		$0.2V_{DD}+1\text{V}$		V <sub>DD</sub>	V	Full V <sub>DD</sub> range <sup>(5)</sup>
		$0.85 V_{DD}$		V <sub>DD</sub>	V	
		$0.85 V_{DD}$		V <sub>DD</sub>	V	EXTRC option only <sup>(4)</sup>
		$0.7 V_{DD}$		V <sub>DD</sub>	V	XT and LP options
<b>Input Leakage Current</b> <sup>(2,3)</sup> I/O ports	I <sub>IL</sub>	-1	0.5	+1	μA	For $V_{DD} \leq 5.5\text{V}$ $V_{SS} \leq V_{PIN} \leq V_{DD}$ , Pin at hi-impedance
		20	130	250	μA	$V_{PIN} = V_{SS} + 0.25\text{V}$ <sup>(2)</sup>
			0.5	+5	μA	$V_{PIN} = V_{DD}$
			0.5	+3	μA	$V_{SS} \leq V_{PIN} \leq V_{DD}$ , XT and LP options
$\overline{\text{MCLR}}$		20	130	250	μA	
OSC1		-3	0.5	+3	μA	
<b>Output Low Voltage</b> I/O ports	V <sub>OL</sub>			0.6	V	I <sub>OL</sub> = 8.7 mA, V <sub>DD</sub> = 4.5V
<b>Output High Voltage</b> <sup>(3,4)</sup> I/O ports	V <sub>OH</sub>	$V_{DD} - 0.7$			V	I <sub>OH</sub> = -5.4 mA, V <sub>DD</sub> = 4.5V

\* These parameters are characterized but not tested.

Note 1: Data in the Typical ("Typ") column is based on characterization results at 25°C. This data is for design guidance only and is not tested.

- 2: The leakage current on the  $\overline{\text{MCLR}}/V_{PP}$  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 voltage.
- 3: Negative current is defined as coming out of the pin.
- 4: For PIC12C5XX devices, the OSC1/CLKIN pin is a Schmitt Trigger input. It is not recommended that the PIC12C5XX be driven with external clock in RC mode.
- 5: The user may use the better of the two specifications.

# PIC12C5XX

## 10.3 Timing Parameter Symbology and Load Conditions

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

1. TppS2ppS
2. TppS

<b>T</b>			
F	Frequency	T	Time

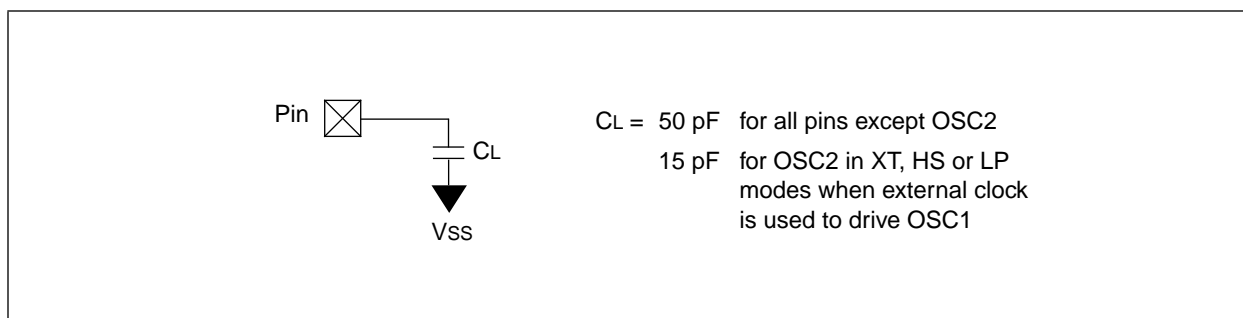
Lowercase subscripts (pp) and their meanings:

<b>pp</b>			
2	to	mc	MCLR
ck	CLKOUT	osc	oscillator
cy	cycle time	os	OSC1
drt	device reset timer	t0	T0CKI
io	I/O port	wdt	watchdog timer

Uppercase letters and their meanings:

<b>S</b>			
F	Fall	P	Period
H	High	R	Rise
I	Invalid (Hi-impedance)	V	Valid
L	Low	Z	Hi-impedance

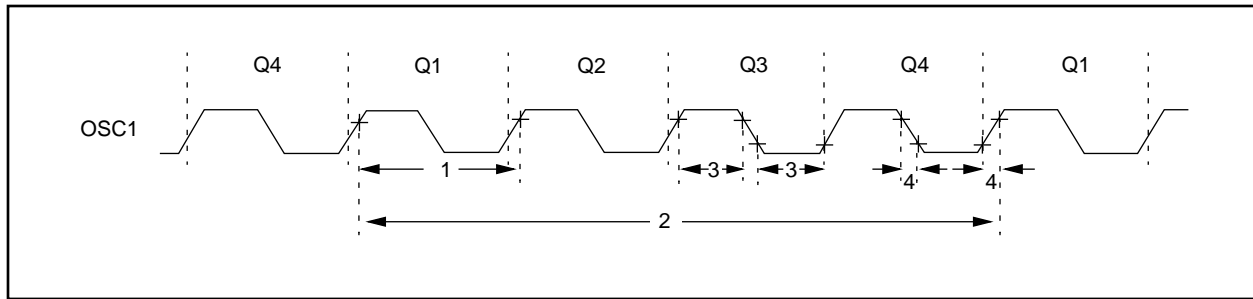
**FIGURE 10-1: LOAD CONDITIONS - PIC12C5XX**





## 10.4 Timing Diagrams and Specifications

**FIGURE 10-2: EXTERNAL CLOCK TIMING - PIC12C5XX**



**TABLE 10-1: EXTERNAL CLOCK TIMING REQUIREMENTS - PIC12C5XX**

AC Characteristics		Standard Operating Conditions (unless otherwise specified)					
		Operating Temperature $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$ (commercial), $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ (industrial), $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ (extended)					
		Operating Voltage $V_{DD}$ range is described in Section 10.1					
Parameter No.	Sym	Characteristic	Min	Typ <sup>(1)</sup>	Max	Units	Conditions
	Fosc	External CLKIN Frequency <sup>(2)</sup>	DC	—	4	MHz	EXTRC osc mode
			DC	—	4	MHz	XT osc mode
			DC	—	200	kHz	LP osc mode
	Oscillator Frequency <sup>(2)</sup>	DC	—	4	MHz	EXTRC osc mode	
		0.1	—	4	MHz	XT osc mode	
		DC	—	200	kHz	LP osc mode	
1	Tosc	External CLKIN Period <sup>(2)</sup>	250	—	—	ns	EXTRC osc mode
			250	—	—	ns	XT osc mode
			5	—	—	ms	LP osc mode
	Oscillator Period <sup>(2)</sup>	250	—	—	ns	EXTRC osc mode	
		250	—	10,000	ns	XT osc mode	
		5	—	—	ms	LP osc mode	
2	Tcy	Instruction Cycle Time <sup>(3)</sup>	—	4/Fosc	—	—	
3	TosL, TosH	Clock in (OSC1) Low or High Time	50*	—	—	ns	XT oscillator
			2*	—	—	ms	LP oscillator
4	TosR, TosF	Clock in (OSC1) Rise or Fall Time	—	—	25*	ns	XT oscillator
			—	—	50*	ns	LP oscillator

\* These parameters are characterized but not tested.

Note 1: Data in the Typical ("Typ") column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

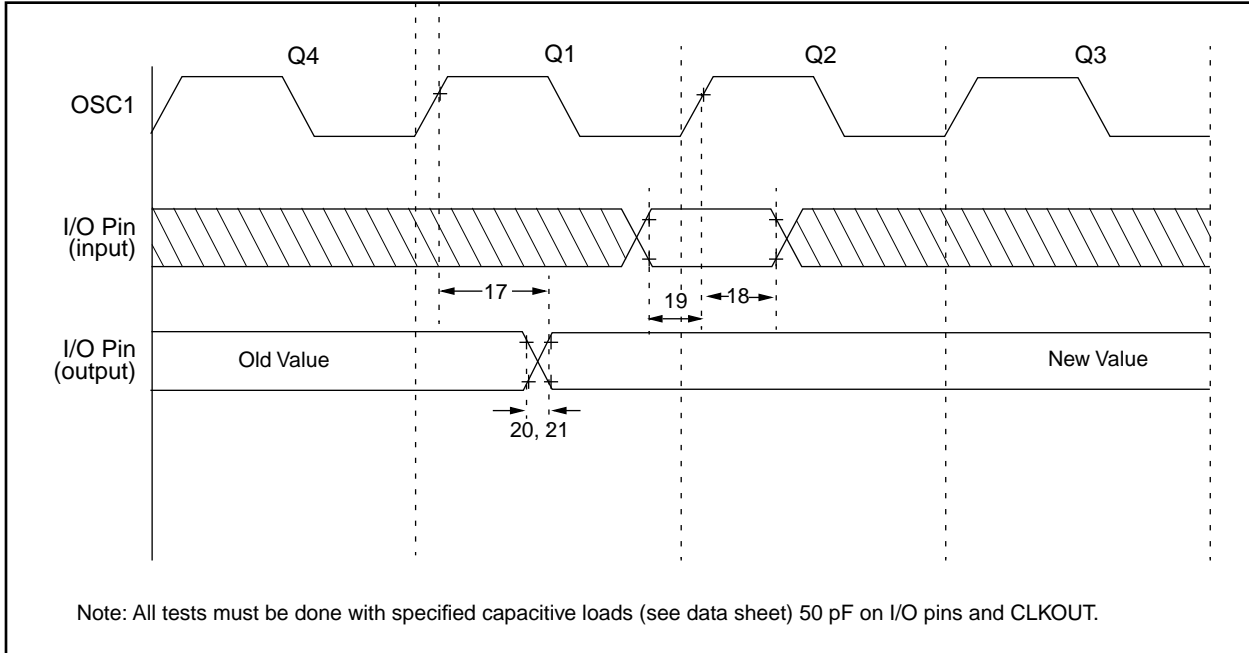
2: All specified values are based on characterization data for that particular oscillator type under standard operating conditions with the device executing code. Exceeding these specified limits may result in an unstable oscillator operation and/or higher than expected current consumption.

When an external clock input is used, the "max" cycle time limit is "DC" (no clock) for all devices.

3: Instruction cycle period (Tcy) equals four times the input oscillator time base period.

# PIC12C5XX

**FIGURE 10-3: I/O TIMING - PIC12C5XX**



**TABLE 10-2: TIMING REQUIREMENTS - PIC12C5XX**

AC Characteristics		Standard Operating Conditions (unless otherwise specified)				
		Operating Temperature $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$ (commercial) $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ (industrial) $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ (extended)				
		Operating Voltage $V_{DD}$ range is described in Section 10.1				
Parameter No.	Sym	Characteristic	Min	Typ <sup>(1)</sup>	Max	Units
17	TosH2ioV	OSC1 $\uparrow$ (Q1 cycle) to Port out valid <sup>(3)</sup>	—	—	100*	ns
18	TosH2ioI	OSC1 $\uparrow$ (Q2 cycle) to Port input invalid (I/O in hold time)	TBD	—	—	ns
19	TioV2osH	Port input valid to OSC1 $\uparrow$ (I/O in setup time)	TBD	—	—	ns
20	TioR	Port output rise time <sup>(3)</sup>	—	10	25**	ns
21	TioF	Port output fall time <sup>(3)</sup>	—	10	25**	ns

\* These parameters are characterized but not tested.

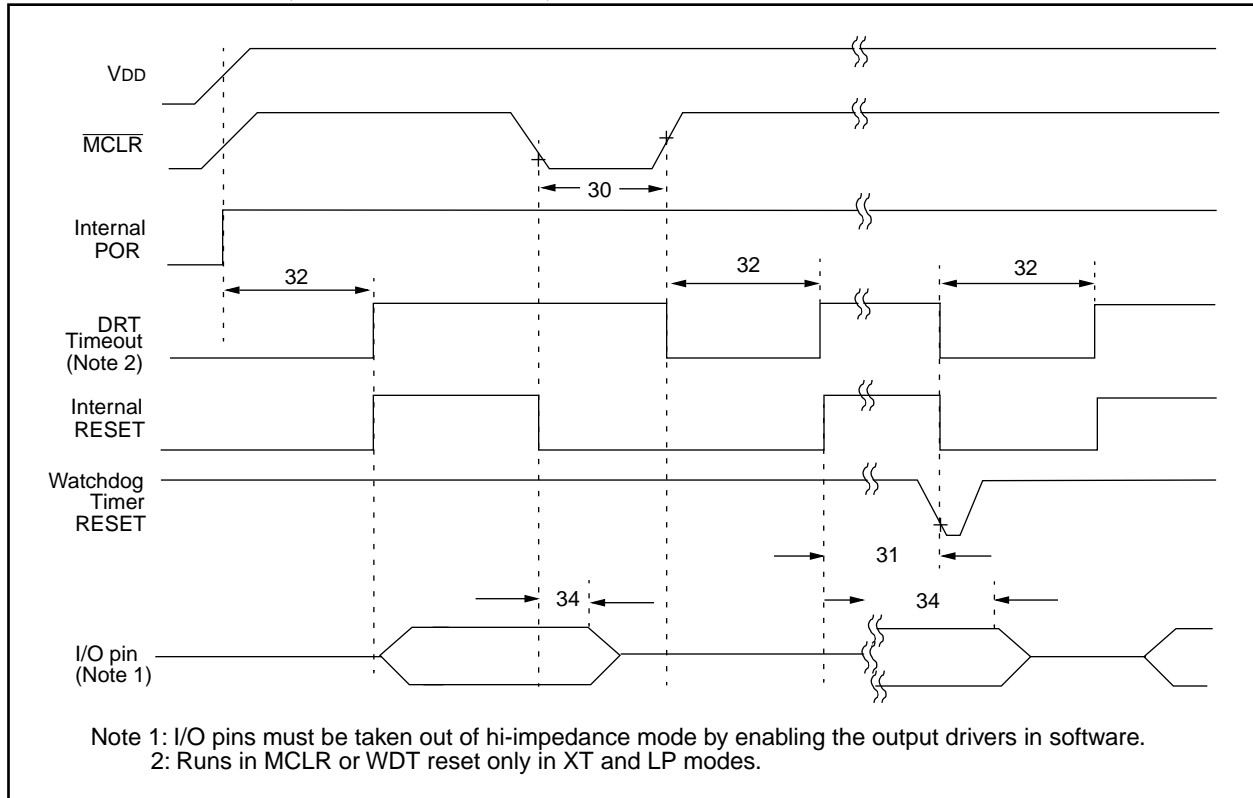
\*\* These parameters are design targets and are not tested. No characterization data available at this time.

Note 1: Data in the Typical ("Typ") column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

2: Measurements are taken in EXTRC mode.

3: See Figure 10-1 for loading conditions.

**FIGURE 10-4: RESET, WATCHDOG TIMER, AND DEVICE RESET TIMER TIMING - PIC12C5XX**



**TABLE 10-3: RESET, WATCHDOG TIMER, AND DEVICE RESET TIMER - PIC12C5XX**

AC Characteristics Standard Operating Conditions (unless otherwise specified)							
		Operating Temperature		0°C ≤ TA ≤ +70°C (commercial) -40°C ≤ TA ≤ +85°C (industrial) -40°C ≤ TA ≤ +125°C (extended)			
		Operating Voltage VDD range is described in Section 10.1					
Parameter No.	Sym	Characteristic	Min	Typ <sup>(1)</sup>	Max	Units	Conditions
30	Tmcl	MCLR Pulse Width (low)	2000*	—	—	ns	VDD = 5 V
31	Twdt	Watchdog Timer Time-out Period (No Prescaler)	9*	18*	30*	ms	VDD = 5 V (Commercial)
32	TDRT	Device Reset Timer Period <sup>(2)</sup>	9*	18*	30*	ms	VDD = 5 V (Commercial)
34	Tioz	I/O Hi-impedance from MCLR Low	—	—	2000*	ns	

\* These parameters are characterized but not tested.

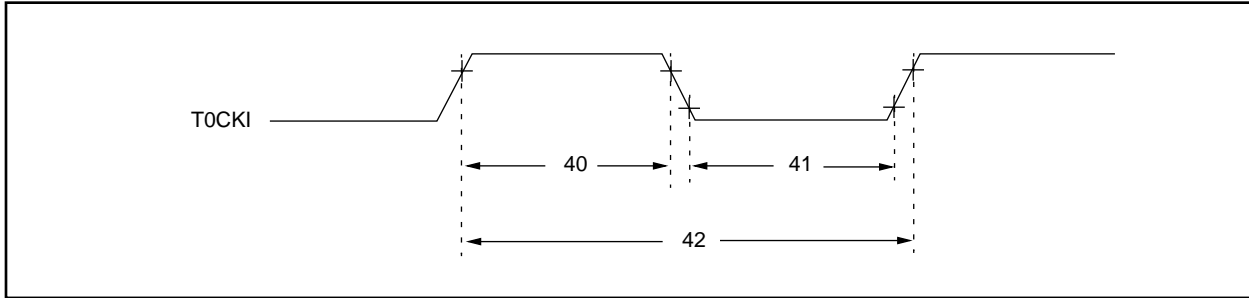
Note 1: Data in the Typical ("Typ") column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**TABLE 10-4: DRT (DEVICE RESET TIMER PERIOD)**

Oscillator Configuration	POR Reset	Subsequent Resets
IntRC & ExtRC	18 ms (typical)	300 μs (typical)
XT & LP	18 ms (typical)	18 ms (typical)

# PIC12C5XX

**FIGURE 10-5: TIMER0 CLOCK TIMINGS - PIC12C5XX**



**TABLE 10-5: TIMER0 CLOCK REQUIREMENTS - PIC12C5XX**

AC Characteristics		Standard Operating Conditions (unless otherwise specified)					
		Operating Temperature 0°C ≤ TA ≤ +70°C (commercial) -40°C ≤ TA ≤ +85°C (industrial) -40°C ≤ TA ≤ +125°C (extended)					
		Operating Voltage VDD range is described in Section 10.1.					
Parameter No.	Sym	Characteristic	Min	Typ <sup>(1)</sup>	Max	Units	Conditions
40	Tt0H	TOCKI High Pulse Width - No Prescaler	0.5 Tcy + 20*	—	—	ns	
		- With Prescaler	10*	—	—	ns	
41	Tt0L	TOCKI Low Pulse Width - No Prescaler	0.5 Tcy + 20*	—	—	ns	
		- With Prescaler	10*	—	—	ns	
42	Tt0P	TOCKI Period	20 or $\frac{Tcy + 40}{N}$ *	—	—	ns	Whichever is greater. N = Prescale Value (1, 2, 4, ..., 256)

\* These parameters are characterized but not tested.

Note 1: Data in the Typical ("Typ") column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**TABLE 10-6: PULL-UP RESISTOR RANGES**

VDD (Volts)	Temperature (°C)	Min	Typ	Max	Units
<b>GP0/GP1</b>					
2.5	-40	50K	62K	84K	Ω
	25	63K	73K	84K	Ω
	85	63K	80K	84K	Ω
	125	63K	81K	84K	Ω
5.5	-40	19K	22K	27K	Ω
	25	25K	27K	31K	Ω
	85	28K	33K	40K	Ω
	125	31K	36K	43K	Ω
<b>GP3</b>					
2.5	-40	490K	710K	965K	Ω
	25	610K	890K	1.2M	Ω
	85	769K	1.1M	1.5M	Ω
	125	810K	1.2M	1.6M	Ω
5.5	-40	310K	410K	510K	Ω
	25	360K	470K	580K	Ω
	85	430K	550K	670K	Ω
	125	465K	585K	715K	Ω

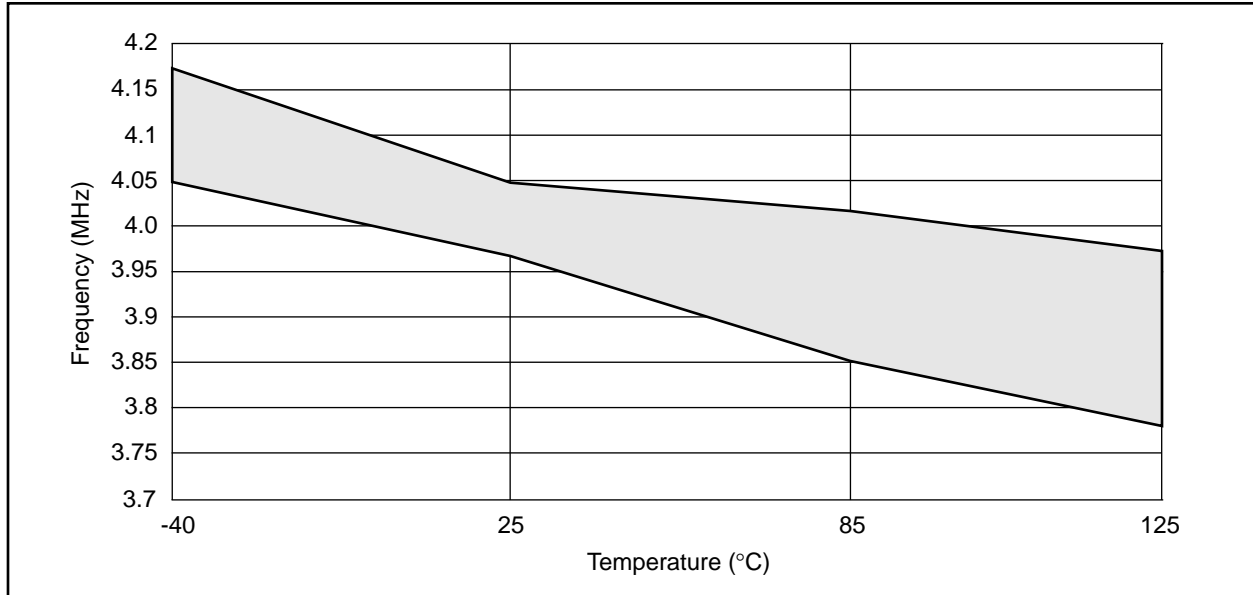
\* These parameters are characterized but not tested.

## 11.0 DC AND AC CHARACTERISTICS - PIC12C5XX

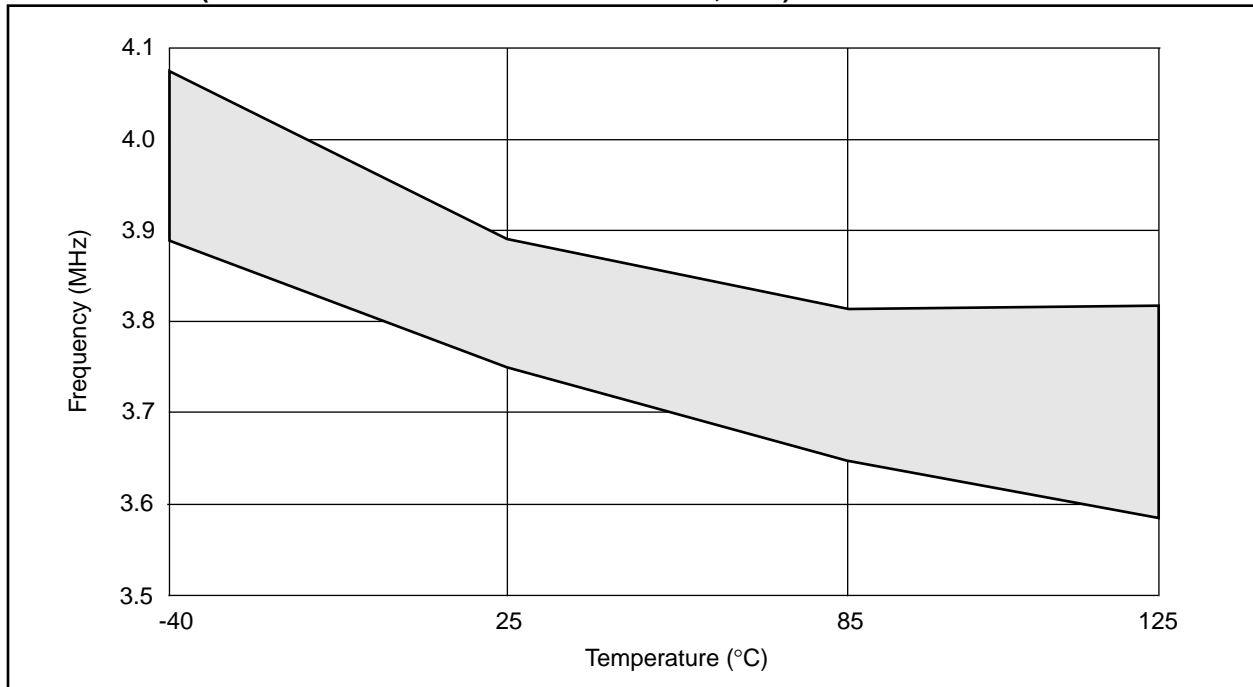
The graphs and tables provided in this section are for design guidance and are not tested or guaranteed. In some graphs or tables the data presented are outside specified operating range (e.g., outside specified  $V_{DD}$  range). This is for information only and devices will operate properly only within 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 +  $3\sigma$ ) and (mean -  $3\sigma$ ) respectively, where  $\sigma$  is standard deviation.

**FIGURE 11-1: CALIBRATED INTERNAL RC FREQUENCY RANGE VS. TEMPERATURE ( $V_{DD} = 5.0V$ )  
(INTERNAL RC IS CALIBRATED TO 25°C, 5.0V)**

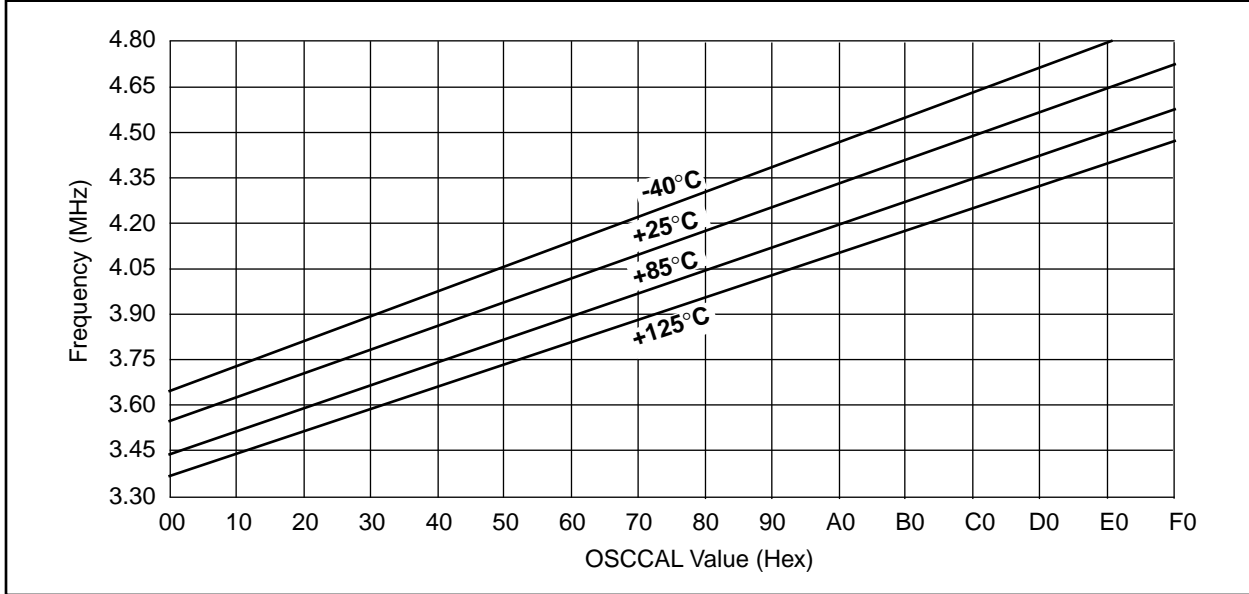


**FIGURE 11-2: CALIBRATED INTERNAL RC FREQUENCY RANGE VS. TEMPERATURE ( $V_{DD} = 3.0V$ )  
(INTERNAL RC IS CALIBRATED TO 25°C, 5.0V)**

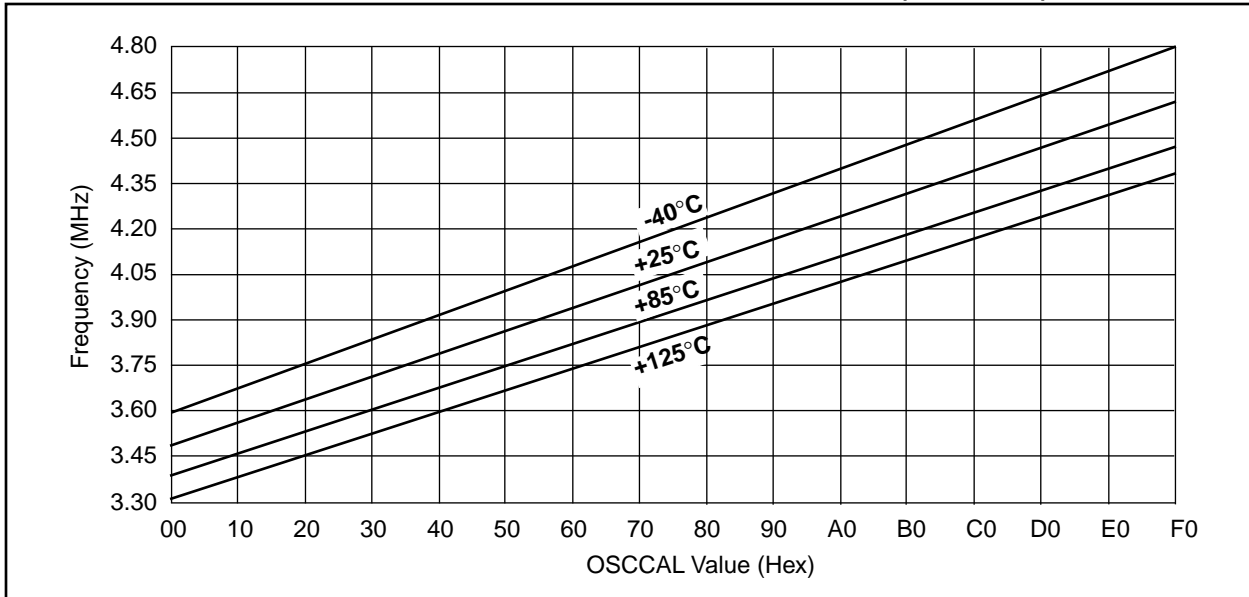


# PIC12C5XX

**FIGURE 11-3: INTERNAL RC FREQUENCY VS. CALIBRATION VALUE (V<sub>DD</sub> = 5.5V)**



**FIGURE 11-4: INTERNAL RC FREQUENCY VS. CALIBRATION VALUE (V<sub>DD</sub> = 3.5V)**

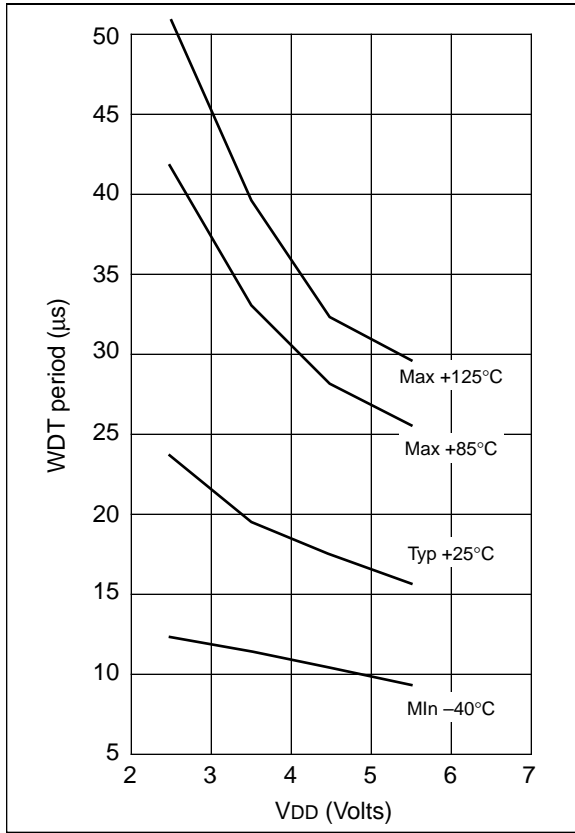


**TABLE 11-1: DYNAMIC I<sub>DD</sub> (TYPICAL) - WDT ENABLED, 25°C**

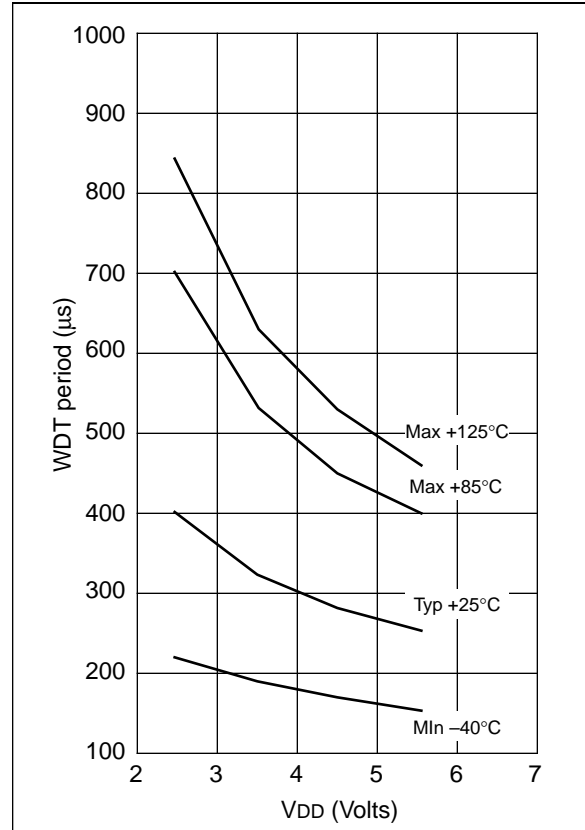
Oscillator	Frequency	V <sub>DD</sub> = 2.5V	V <sub>DD</sub> = 5.5V
External RC	4 MHz	250 μA*	620 μA*
Internal RC	4 MHz	420 μA	1.1 mA
XT	4 MHz	251 μA	775 μA
LP	32 KHz	7 μA	37 μA

\*Does not include current through external R&C.

**FIGURE 11-5: WDT TIMER TIME-OUT PERIOD vs. VDD**

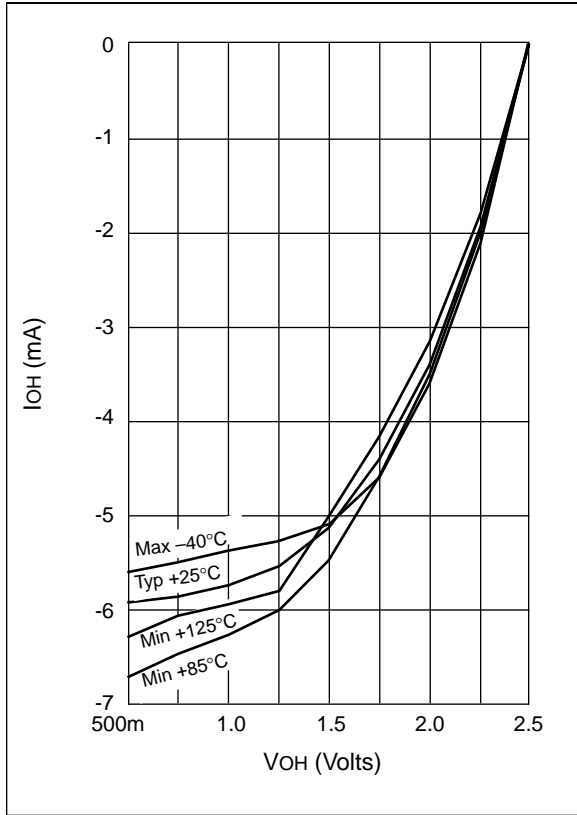


**FIGURE 11-6: SHORT DRT PERIOD VS. VDD**

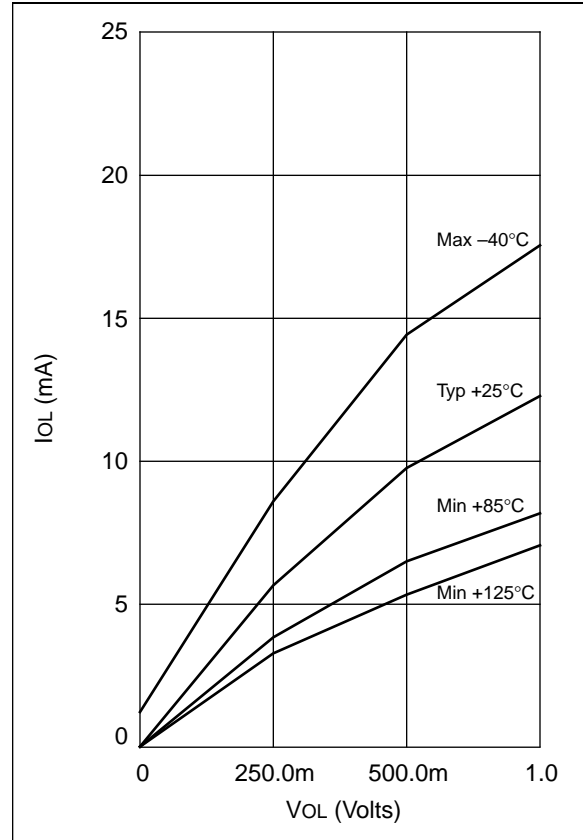


# PIC12C5XX

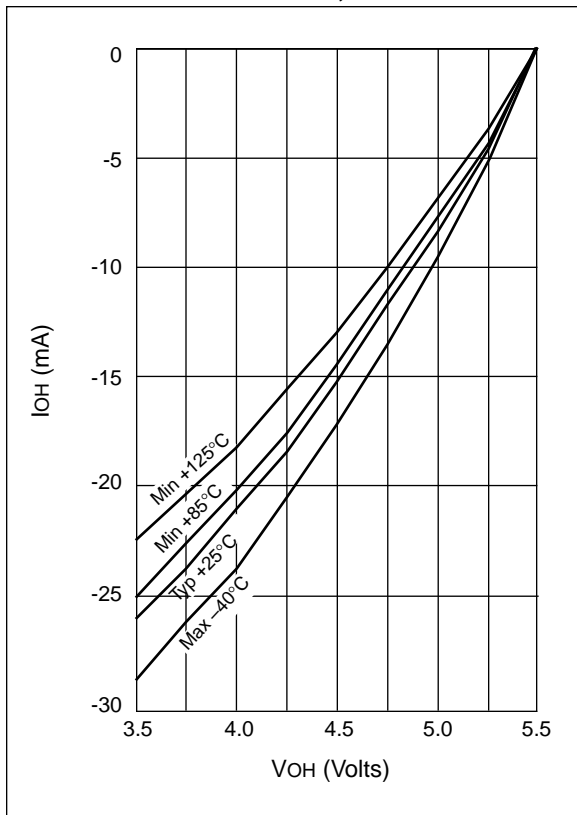
**FIGURE 11-7:  $I_{OH}$  vs.  $V_{OH}$ ,  $V_{DD} = 2.5\text{ V}$**



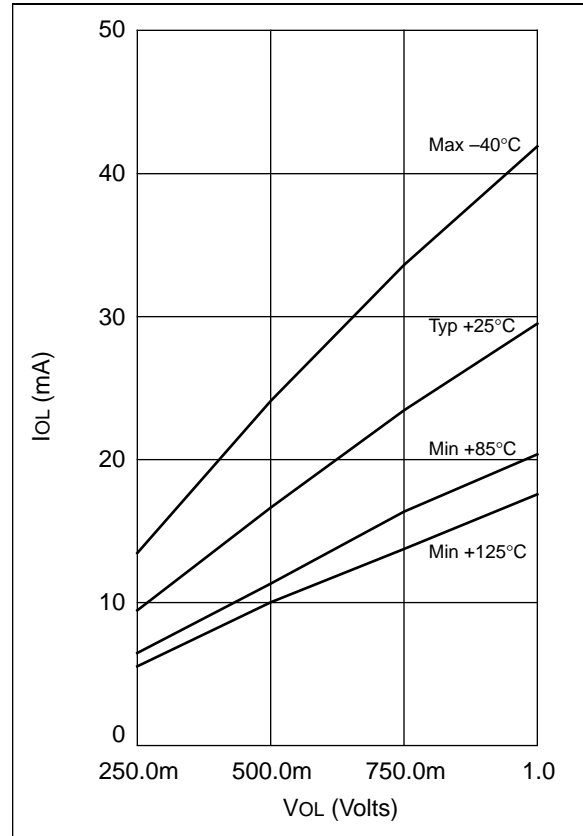
**FIGURE 11-9:  $I_{OL}$  vs.  $V_{OL}$ ,  $V_{DD} = 2.5\text{ V}$**



**FIGURE 11-8:  $I_{OH}$  vs.  $V_{OH}$ ,  $V_{DD} = 5.5\text{ V}$**



**FIGURE 11-10:  $I_{OL}$  vs.  $V_{OL}$ ,  $V_{DD} = 5.5\text{ V}$**





## 12.0 PACKAGING INFORMATION

### 12.1 Package Marking Information

#### 8-Lead PDIP (300 mil)



#### Example



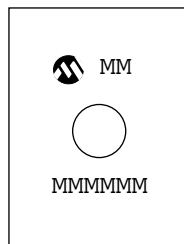
#### 8-Lead SOIC (208 mil)



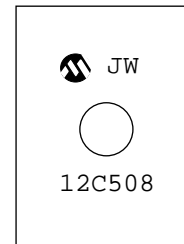
#### Example



#### 8-Lead Windowed Ceramic Side Brazed (300 mil)



#### Example



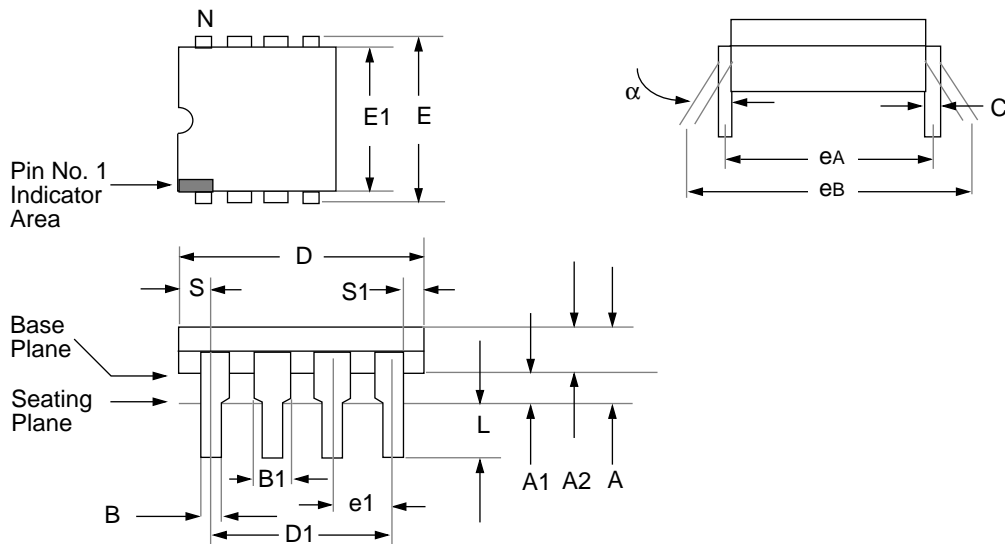
<b>Legend:</b> MM...M	Microchip part number information
XX...X	Customer specific information*
AA	Year code (last 2 digits of calendar year)
BB	Week code (week of January 1 is week '01')
C	Facility code of the plant at which wafer is manufactured
	C = Chandler, Arizona, U.S.A.,
	S = Tempe, Arizona, U.S.A.
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.

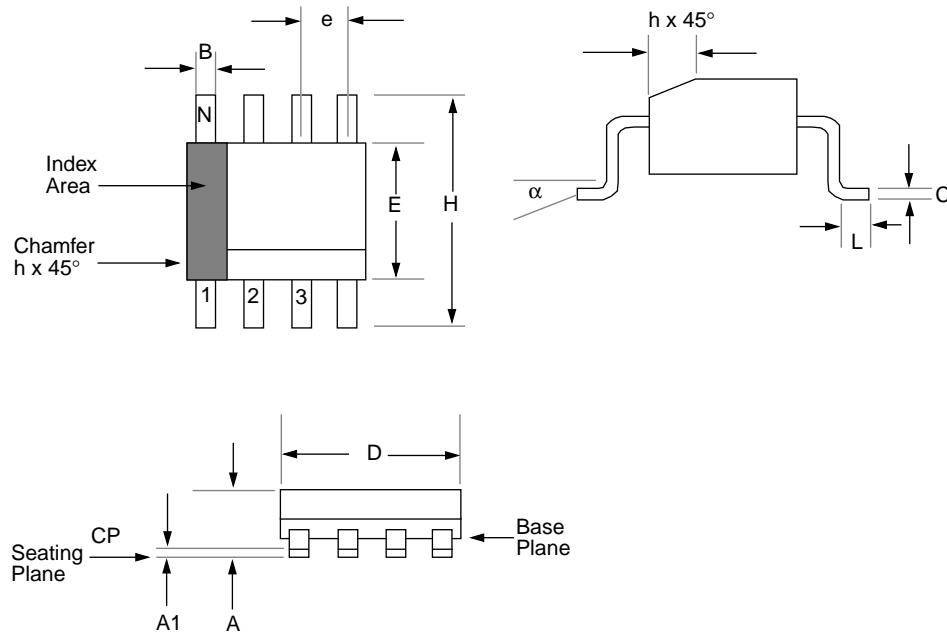
# PIC12C5XX

## 12.2 8-Lead Plastic Dual In-line (300 mil)



Package Group: Plastic Dual In-Line (PLA)						
Symbol	Millimeters			Inches		
	Min	Max	Notes	Min	Max	Notes
$\alpha$	0°	10°		0°	10°	
A	–	4.064		–	0.160	
A1	0.381	–		0.015	–	
A2	3.048	3.810		0.120	0.150	
B	0.355	0.559		0.014	0.022	
B1	1.397	1.651		0.055	0.065	
C	0.203	0.381	Typical	0.008	0.015	Typical
D	9.017	10.922		0.355	0.430	
D1	7.620	7.620	Reference	0.300	0.300	Reference
E	7.620	8.255		0.300	0.325	
E1	6.096	7.112		0.240	0.280	
e1	2.489	2.591	Typical	0.098	0.102	Typical
eA	7.620	7.620	Reference	0.300	0.300	Reference
eB	7.874	9.906		0.310	0.390	
L	3.048	3.556		0.120	0.140	
N	8	8		8	8	
S	0.889	–		0.035	–	
S1	0.254	–		0.010	–	

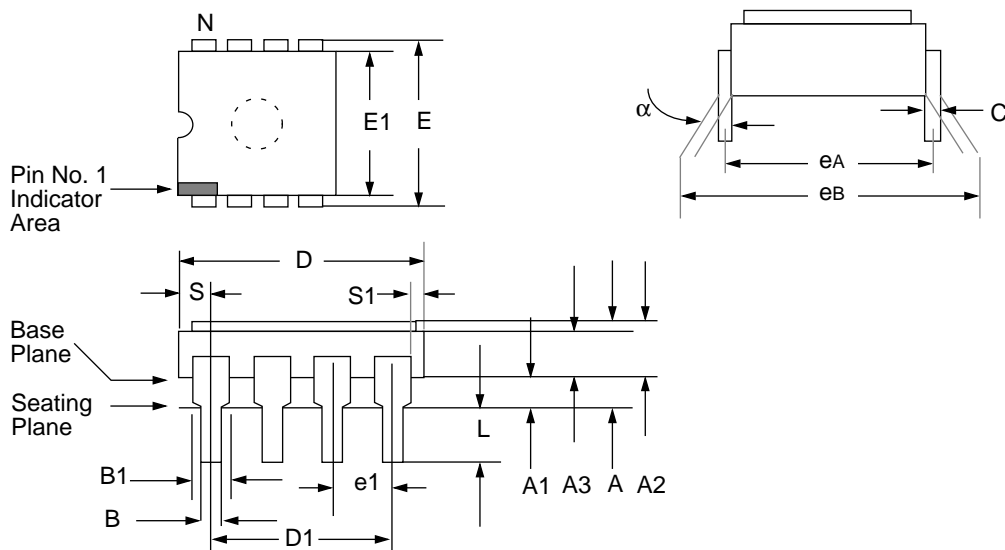
## 12.3 8-Lead Plastic Surface Mount (SOIC - Medium, 208 mil Body)



Package Group: Plastic SOIC (SM)						
Symbol	Millimeters			Inches		
	Min	Max	Notes	Min	Max	Notes
$\alpha$	0°	8°		0°	8°	
A	1.778	2.00		0.070	0.079	
A1	0.101	0.249		0.004	0.010	
B	0.355	0.483		0.014	0.019	
C	0.190	0.249		0.007	0.010	
D	5.080	5.334		0.200	0.210	
E	5.156	5.411		0.203	0.213	
e	1.270	1.270	Reference	0.050	0.050	Reference
H*	7.670	8.103		0.302	0.319	
h	0.381	0.762		0.015	0.030	
L	0.508	1.016		0.020	0.040	
N	14	14		14	14	
CP	—	0.102		—	0.004	

# PIC12C5XX

## 12.4 8-Lead Ceramic Side Brazed Dual In-Line with Window (JW) (300 mil)



Package Group: Ceramic Side Brazed Dual In-Line (CER)

Symbol	Millimeters			Inches		
	Min	Max	Notes	Min	Max	Notes
$\alpha$	0°	10°		0°	10°	
A	3.937	5.030		0.155	0.198	
A1	0.635	1.143		0.025	0.045	
A2	2.921	3.429		0.115	0.135	
A3	1.778	2.413		0.070	0.095	
B	0.406	0.508		0.016	0.020	
B1	1.371	1.371	Typical	0.054	0.054	Typical
C	0.228	0.305	Typical	0.009	0.012	Typical
D	13.004	13.412		0.512	0.528	
D1	7.416	7.824	BSC	0.292	0.308	BSC
E	7.569	8.230		0.298	0.324	
E1	7.112	7.620		0.280	0.300	
e1	2.540	2.540	Typical	0.100	0.100	Typical
eA	7.620	7.620	BSC	0.300	0.300	BSC
eB	7.620	9.652		0.300	0.380	
L	3.302	4.064		0.130	0.160	
S	2.540	3.048		0.100	0.120	
S1	0.127	—		0.005	—	

## INDEX

### A

ALU .....	7
Applications.....	3
Architectural Overview .....	7
Assembler .....	50

### B

Block Diagram	
On-Chip Reset Circuit.....	30
Timer0.....	21
TMR0/WDT Prescaler.....	24
Watchdog Timer.....	33
Brown-Out Protection Circuit .....	34

### C

C Compiler (MP-C) .....	51
Carry .....	7
Clocking Scheme .....	10
Code Protection .....	25, 35
Configuration Bits.....	25
Configuration Word	
PIC12C508/509 .....	25

### D

Development Support .....	49
Development Tools .....	49
Device Varieties .....	5
Digit Carry .....	7

### F

Features.....	1
FSR.....	17
Fuzzy Logic Dev. System ( <i>fuzzyTECH</i> <sup>®</sup> -MP).....	49, 51

### I

I/O Interfacing .....	19
I/O Ports.....	19
I/O Programming Considerations.....	20
ID Locations .....	25, 35
INDF.....	17
Indirect Data Addressing.....	17
Instruction Cycle .....	10
Instruction Flow/Pipelining .....	10
Instruction Set Summary.....	38
Internal Weak Pull-ups.....	15, 19

### L

Loading of PC .....	16
---------------------	----

### M

Memory Organization.....	11
Data Memory .....	12
Program Memory .....	11
MPASM Assembler.....	49, 50
MP-C C Compiler .....	51
MPSIM Software Simulator.....	49, 51

### O

One-Time-Programmable (OTP) Devices.....	5
OPTION Register.....	15
OSC Selection .....	25
Oscillator Configurations.....	26
Oscillator Types	
INTRL.....	26
LP.....	26
EXT RC.....	26
XT .....	26

### P

Package Marking Information .....	65
Packaging Information .....	65
PC .....	16

PICDEM-1 Low-Cost PIC16/17 Demo Board .....	49, 50
PICDEM-2 Low-Cost PIC16CXX Demo Board.....	49, 50
PICDEM-3 Low-Cost PIC16C9XXX Demo Board .....	50
PICMASTER <sup>®</sup> RT In-Circuit Emulator .....	49
PICSTART <sup>®</sup> Low-Cost Development System.....	49
Pin Compatible Devices.....	76
POR	

Device Reset Timer (DRT) .....	25, 32
$\overline{PD}$ .....	34
Power-On Reset (POR) .....	25
TO.....	34

GPIO.....	19
Power-Down Mode .....	35
Prescaler .....	24
PRO MATE <sup>®</sup> Universal Programmer.....	49
Program Counter .....	16

### Q

Q cycles.....	10
---------------	----

### R

RC Oscillator .....	27
Read Modify Write .....	20
Register File Map	
PIC12C508/509 .....	12

### Registers

Special Function .....	13
------------------------	----

Reset .....	25
Reset on Brown-Out .....	34

### S

SLEEP .....	25, 35
Software Simulator (MPSIM) .....	51
Special Features of the CPU.....	25
Special Function Registers.....	13
Stack.....	16
STATUS .....	7
STATUS Register .....	14

### T

Timer0	
Switching Prescaler Assignment .....	24
Timer0 .....	21
Timer0 (TMR0) Module .....	21
TMR0 with External Clock .....	23
Timing Diagrams and Specifications .....	57
Timing Parameter Symbolology and Load Conditions .....	56
TRIS Registers .....	19

### W

Wake-up on Change.....	19
Wake-up from SLEEP .....	35
Watchdog Timer (WDT).....	25, 32
Period .....	33
Programming Considerations .....	33

### Z

Zero bit .....	7
----------------	---

# PIC12C5XX

## LIST OF EXAMPLES

Example 3-1: Instruction Pipeline Flow .....	10
Example 4-1: Indirect Addressing .....	17
Example 4-2: How To Clear RAM Using Indirect Addressing .....	17
Example 5-1: Read-Modify-Write Instructions on an I/O Port .....	20
Example 6-1: Changing Prescaler (Timer0→WDT) .....	24
Example 6-2: Changing Prescaler (WDT→Timer0) .....	24

## LIST OF FIGURES

Figure 3-1: PIC12C5XX Block Diagram .....	8
Figure 3-2: Clock/Instruction Cycle .....	10
Figure 4-1: Program Memory Map and Stack for the PIC12C5XX .....	11
Figure 4-2: PIC12C508 Register File Map .....	12
Figure 4-3: PIC12C509 Register File Map .....	12
Figure 4-4: STATUS Register (Address:03h) .....	14
Figure 4-5: OPTION Register .....	15
Figure 4-6: Loading of PC Branch Instructions - PIC12C508/C509 .....	16
Figure 4-7: Direct/Indirect Addressing .....	17
Figure 5-1: Equivalent Circuit for a Single I/O Pin .....	19
Figure 5-2: Successive I/O Operation .....	20
Figure 6-1: Timer0 Block Diagram .....	21
Figure 6-2: Timer0 Timing: Internal Clock/No Prescale .....	22
Figure 6-3: Timer0 Timing: Internal Clock/Prescale 1:2 .....	22
Figure 6-4: Timer0 Timing With External Clock .....	23
Figure 6-5: Block Diagram of the Timer0/WDT Prescaler .....	24
Figure 7-1: Configuration Word for PIC12C508 or PIC12C509 .....	25
Figure 7-2: Crystal Operation (or Ceramic Resonator) (XT or LP OSC Configuration) .....	26
Figure 7-3: External Clock Input Operation (XT or LP OSC Configuration) .....	26
Figure 7-4: External Parallel Resonant Crystal Oscillator Circuit .....	27
Figure 7-5: External Series Resonant Crystal Oscillator Circuit .....	27
Figure 7-6: External RC Oscillator Mode .....	27
Figure 7-7: MCLR Select .....	29
Figure 7-8: Simplified Block Diagram of On-Chip Reset Circuit .....	30
Figure 7-9: Time-Out Sequence on Power-Up (MCLR Pulled Low) .....	31
Figure 7-10: Time-Out Sequence on Power-Up (MCLR Tied to VDD): Fast VDD Rise Time .....	31
Figure 7-11: Time-Out Sequence on Power-Up (MCLR Tied to VDD): Slow VDD Rise Time .....	31
Figure 7-12: Watchdog Timer Block Diagram .....	33
Figure 7-13: Brown-Out Protection Circuit 1 .....	34
Figure 7-14: Brown-Out Protection Circuit 2 .....	34
Figure 7-15: Typical In-Circuit Serial Programming Connection .....	36
Figure 8-1: General Format for Instructions .....	37
Figure 10-1: Load Conditions - PIC12C5XX .....	56
Figure 10-2: External Clock Timing - PIC12C5XX .....	57
Figure 10-3: I/O Timing - PIC12C5XX .....	58
Figure 10-4: Reset, Watchdog Timer, and Device Reset Timer Timing - PIC12C5XX .....	59
Figure 10-5: Timer0 Clock Timings - PIC12C5XX .....	60

Figure 11-1: Calibrated Internal RC Frequency Range vs. Temperature (VDD = 5.0V) (Internal RC is Calibrated to 25°C, 5.0V) ..	61
Figure 11-2: Calibrated Internal RC Frequency Range vs. Temperature (VDD = 3.0V) (Internal RC is Calibrated to 25°C, 5.0V) ..	61
Figure 11-3: Internal RC Frequency vs. Calibration Value (VDD = 5.5V) .....	62
Figure 11-4: Internal RC Frequency vs. Calibration Value (VDD = 3.5V) .....	62
Figure 11-5: WDT Timer Time-out Period vs. VDD .....	63
Figure 11-6: Short DRT Period vs. VDD .....	63
Figure 11-7: IOH vs. VOH, VDD = 2.5 V .....	64
Figure 11-8: IOH vs. VOH, VDD = 5.5 V .....	64
Figure 11-9: IOL vs. VOL, VDD = 2.5 V .....	64
Figure 11-10: IOL vs. VOL, VDD = 5.5 V .....	64

## LIST OF TABLES

Table 1-1: PIC12C5XX Family of Devices .....	4
Table 3-1: PIC12C5XX Pinout Description .....	9
Table 4-1: Special Function Register Summary .....	13
Table 5-1: Summary of Port Registers .....	19
Table 6-1: Registers Associated With Timer0 .....	22
Table 7-1: Capacitor Selection For Ceramic Resonators - PIC12C5XX .....	26
Table 7-2: Capacitor Selection For Crystal Oscillator - PIC12C5XX .....	26
Table 7-3: Reset Conditions For Registers .....	28
Table 7-4: Reset Condition For Special Registers .....	29
Table 7-5: DRT (Device Reset Timer Period) .....	32
Table 7-6: Summary of Registers Associated with the Watchdog Timer .....	33
Table 7-7: $\overline{TO}/\overline{PD}/\overline{GPWUF}$ Status After Reset .....	34
Table 7-8: Events Affecting $\overline{TO}/\overline{PD}$ Status Bits .....	34
Table 8-1: OPCODE Field Descriptions .....	37
Table 8-2: Instruction Set Summary .....	38
Table 9-1: Development Tools From Microchip .....	52
Table 10-1: External Clock Timing Requirements - PIC12C5XX .....	57
Table 10-2: Timing Requirements - PIC12C5XX .....	58
Table 10-3: Reset, Watchdog Timer, and Device Reset Timer - PIC12C5XX .....	59
Table 10-4: DRT (Device Reset Timer Period) .....	59
Table 10-5: Timer0 Clock Requirements - PIC12C5XX .....	60
Table 10-6: Pull-up Resistor Ranges .....	60
Table 11-1: Dynamic IDD (Typical) - WDT Enabled, 25°C .....	62

## 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 microcontroller 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.

**Trademarks:** The Microchip name, logo, PIC, PICSTART, PICMASTER, and are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries. *FlexROM*, *MPLAB*, *PRO MATE*, 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.

# PIC12C5XX

---

## 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: **PIC12C5XX**                      Literature Number: **DS40139B**

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?  
 \_\_\_\_\_  
 \_\_\_\_\_



## PIC12C5XX PRODUCT IDENTIFICATION SYSTEM

PART NO.	-XX	X	/XX	XXX		Examples
					<b>Pattern:</b>	Special Requirements
					<b>Package:</b>	SM = 208 mil SOIC P = 300 mil PDIP JW = 300 mil Windowed Ceramic Side Brazed
					<b>Temperature Range:</b>	- = 0°C to +70°C I = -40°C to +85°C E = -40°C to +125°C
					<b>Frequency Range:</b>	04 = 4 MHz
					<b>Device</b>	PIC12C508 PIC12C509 PIC12C508T (Tape & reel for SOIC only) PIC12C509T (Tape & reel for SOIC only)
						a) PIC12C508-04/P Commercial Temp., PDIP Package, 4 MHz, normal VDD limits b) PIC12C508-04I/SM Industrial Temp., SOIC package, 4 MHz, normal VDD limits c) PIC12C509-04I/P Industrial Temp., PDIP package, 4 MHz, normal VDD limits

Please contact your local sales office for exact ordering procedures.

### Sales and Support

Products supported by a preliminary Data Sheet may possibly have an errata sheet describing minor operational differences and recommended workarounds. To determine if an errata sheet exists for a particular device, please contact one of the following:

1. Your local Microchip sales office (see below)
2. The Microchip Corporate Literature Center U.S. FAX: (602) 786-7277
3. The Microchip's Bulletin Board, via your local CompuServe number (CompuServe membership NOT required).

Please specify which device, revision of silicon and Data Sheet (include Literature #) you are using.

For latest version information and upgrade kits for Microchip Development Tools, please call 1-800-755-2345 or 1-602-786-7302.

# PIC12C5XX

---

NOTES:

NOTES:

---

---

# WORLDWIDE SALES & SERVICE

---

---

## AMERICAS

### Corporate Office

Microchip Technology Inc.  
2355 West Chandler Blvd.  
Chandler, AZ 85224-6199  
Tel: 602-786-7200 Fax: 602-786-7277  
Technical Support: 602 786-7627  
Web: <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.  
14651 Dallas Parkway, Suite 816  
Dallas, TX 75240-8809  
Tel: 972-991-7177 Fax: 972-991-8588

### Dayton

Microchip Technology Inc.  
Two Prestige Place, Suite 150  
Miamisburg, OH 45342  
Tel: 937-291-1654 Fax: 937-291-9175

### Los Angeles

Microchip Technology Inc.  
18201 Von Karman, Suite 1090  
Irvine, CA 92612  
Tel: 714-263-1888 Fax: 714-263-1338

### New York

Microchip Technology Inc.  
150 Motor Parkway, Suite 416  
Hauppauge, NY 11788  
Tel: 516-273-5305 Fax: 516-273-5335

### San Jose

Microchip Technology Inc.  
2107 North First Street, Suite 590  
San Jose, CA 95131  
Tel: 408-436-7950 Fax: 408-436-7955

### 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  
RM 3801B, Tower Two  
Metroplaza  
223 Hing Fong Road  
Kwai Fong, N.T., Hong Kong  
Tel: 852-2-401-1200 Fax: 852-2-401-3431

### India

Microchip Technology India  
No. 6, Legacy, Convent Road  
Bangalore 560 025, India  
Tel: 91-80-299-4036 Fax: 91-80-559-9840

### 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, Hongjiao District  
Shanghai, PRC 200335  
Tel: 86-21-6275-5700  
Fax: 86 21-6275-5060

### Singapore

Microchip Technology Taiwan  
Singapore Branch  
200 Middle Road  
#10-03 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 2-717-7175 Fax: 886-2-545-0139

## EUROPE

### United Kingdom

Arizona Microchip Technology Ltd.  
Unit 6, The Courtyard  
Meadow Bank, Furlong Road  
Bourne End, Buckinghamshire SL8 5AJ  
Tel: 44-1628-851077 Fax: 44-1628-850259

### France

Arizona Microchip Technology SARL  
Zone Industrielle de la Bonde  
2 Rue du Buisson aux Fraises  
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 Colleone  
Palazzo Taurus 1 V. Le Colleoni 1  
20041 Agrate Brianza  
Milan, Italy  
Tel: 39-39-6899939 Fax: 39-39-6899883

## JAPAN

Microchip Technology Intl. Inc.  
Benex S-1 6F  
3-18-20, Shin Yokohama  
Kohoku-Ku, Yokohama  
Kanagawa 222 Japan  
Tel: 81-4-5471-6166 Fax: 81-4-5471-6122

3/24/97



**MICROCHIP**

All rights reserved. © 1997, Microchip Technology Incorporated, USA.

---

Information contained in this publication regarding device applications and the like is intended through 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. All rights reserved. All other trademarks mentioned herein are the property of their respective companies.